uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales
2017-2018

*Trabajo Fin de Grado*

# Adquisición de datos desde un sensor inercial

---


Alumno:
Daniel Segovia Magaz

Tutor:
Aurelio Ponz Vila



Leganés, Septiembre de 2018
Universidad Carlos III de Madrid
Escuela Politécnica Superior

## Resumen

En el presente trabajo de fin de grado se hace uso de un sensor inercial MPU6050 en conjunto con una placa Arduino UNO para detectar las vibraciones de un vehículo ferroviario con el fin de averiguar el estado de los raíles sobre los que rueda. El sensor estaría instalado en los soportes de las ruedas, de forma que se encuentre lo más cerca posible de la vía y que el chasis del vehículo no absorba las vibraciones. De esta manera se pueden detectar imperfecciones en los raíles. El vehículo en concreto será FerroDron, un dron ferroviario automático cuyo proyecto ha sido llevado a cabo por la Universidad Carlos III de Madrid en conjunto con TECSA, en el que trabajan numerosos sensores en conjunto para las labores de inspección, mantenimiento y vigilancia de vía e infraestructura ferroviaria.

Para llevar a cabo el objetivo, se ha usado la interfaz de Arduino IDE para leer los datos del sensor por pantalla y realizar la correspondiente calibración. Posteriormente, se ha creado un nodo de ROS (Robot Operating System) para publicar un topic con la información proveniente del sensor, y un nodo para que lea los datos desde ROS y los muestre gráficamente. Para esto, se ha usado el paquete rosserial_arduino. Con este paquete, es posible crear un nodo en ROS directamente desde la interfaz para ordenador de Arduino, y que publique los datos recogidos por el sensor en los topics creados.

Por último, en las conclusiones se presenta la Pixhawk como sensor alternativo, que podría cumplir los mismos objetivos establecidos, pero al estar instalada en la plataforma del vehículo ferroviario, el chasis absorbe las vibraciones y podría no suministrar datos reales. Sin embargo, se pueden obtener datos de todo el vehículo, no de cada uno de los raíles, para lo que harían falta dos sensores inerciales como los usados en este trabajo.

## Agradecimientos

En primer lugar, agradecer a mi familia la paciencia que han tenido y que tienen conmigo en estos cuatro años de carrera, en lo que en período de exámenes convivir conmigo no es tarea fácil.

A mis compañeros de la carrera, por confiar en mí y por alentarme en momentos en los que ni yo lo hacía. Sin vosotros no estaría aquí.

A mis amigos del instituto por seguir conmigo después de tantos años. Gracias por ser mi motivación para levantarme cada día.

A mi tutor Aurelio, por toda la ayuda desinteresada que me ha prestado y por darme la oportunidad de realizar este trabajo con él.

A mis compañeros Erasmus, con los que viví la mejor experiencia durante mi estancia en la universidad. Gracias por hacerme crecer como persona y por enseñarme a disfrutar de los pequeños momentos.

A todas las personas que han aportado un granito de arena en mi vida y han hecho de mí la persona que soy. Gracias a todos.

# Tabla de contenido

# Índice de figuras

# Índice de tablas

# 1. Introducción

En este capítulo se exponen los motivos que han conducido a la realización del proyecto, así como los objetivos de este y el marco regulador al que está sujeto. Por último, se plantea la estructura que se ha seguido en esta memoria.

## 1.1. Motivación

Detectar grietas o imperfecciones que no son superficiales en cualquier material es una tarea trabajosa. Se precisan de métodos como los ultrasonidos o los rayos X para llevar a cabo el objetivo. En el ámbito ferroviario, detectar el estado de los raíles antes de que se produzca una falla es complicado si se tuviesen que utilizar métodos como los mencionados anteriormente, sin hablar del coste humano, pues los equipos de mantenimiento ferroviario que deben llevar los correspondientes equipos por el recorrido del tren.

De esta manera, surgió la idea del trabajo. Aprovechando el proyecto FerroDron, que consta de un dron ferroviario automático, se podría instalar un sensor inercial para que tomase datos de los raíles, y haciendo uso de Arduino y ROS para detectar y visualizar las vibraciones del tren e interpretar en qué lugares podría haber defectos.

## 1.2. Objetivos

El objetivo del proyecto es instalar un sensor inercial en los soportes de las ruedas del vehículo ferroviario con el fin de detectar de una manera más sencilla el estado de los raíles, gracias a las vibraciones que produce el propio vehículo sobre las vías. Para ello, como se ha comentado anteriormente, se ha ideado un código en Arduino con este fin y se ha creado varios topic de ROS donde se publican los datos recogidos por el sensor, que se explicará con más detalle en los apartados posteriores.

Al estar instalado en el vehículo FerroDron, que contiene numerosos sensores, ROS nos permitirá integrar el sensor para que trabaje en conjunto con los demás sensores del vehículo (cámaras, láser, inclinómetros, GPS…). Todos los sensores publican con ROS sus datos y el software los combina para obtener datos de la vía.

Resumiendo, los objetivos del proyecto son:

- Elección del sensor inercial.
- Implementación del programa en Arduino con el fin de detectar vibraciones.
- Integración con ROS.
- Visualización de los datos recogidos y correcta interpretación de estos.

## 1.3. Marco regulador

El fin de este trabajo es mejorar la seguridad en las vías ferroviarias anticipando la falla de los raíles. En cuanto a normativa se refiere, la seguridad ferroviaria se recoge en la Orden FOM/167/2015, de 6 de febrero, por la que se regulan las condiciones para la entrada en servicio de subsistemas de carácter estructural, líneas y vehículos ferroviarios [1]. En esta ley emitida por el BOE se recoge que: "antes de prestar servicios de transporte sobre una determinada línea o tramo de la Red Ferroviaria de Interés General, las empresas ferroviarias deberán obtener el certificado de seguridad."

Dado que el vehículo ferroviario no va a transportar ningún pasajero y además su funcionamiento se produce fuera del rango horario del tráfico ferroviario, y que su único objetivo es comprobar el buen estado de las vías, no sería necesario obtener dicho certificado, pero si ayudaría a tener un conocimiento del estado de los raíles para futuras reparaciones. En cualquier caso, esta normativa no es aplicable. Sin embargo, FerroDron debe ser compatible con las circulaciones de los vehículos ferroviarios.

## 1.4. Estructura

Para que la memoria tenga un hilo conductor y se exponga de una manera clara y sencilla se ha organizado de la siguiente manera:

1. **Introducción:** se expone la estructura de la memoria, así como las motivaciones principales para la realización del proyecto y el marco regulador al que está sujeto.
2. **Estado del Arte:** se presenta información previa relacionada con el proyecto, además de las tecnologías existentes que podrían ayudar a cumplir los objetivos propuestos.
3. **Diseño:** se justifican las elecciones de diseño del proyecto, como el microprocesador o el sensor elegido.
4. **Implementación:** se realiza la implementación del diseño previamente justificado: creación de código, integración con ROS, obtención de datos.
5. **Resultados:** interpretación de los datos obtenidos y justificación de estos.
6. **Entorno socio-económico:** se incorpora el presupuesto de la elaboración del proyecto, así como el impacto socio-económico que tendría la aplicación del trabajo sobre el sector.
7. **Conclusiones:** destaca las conclusiones tras la realización del proyecto y se presentan algunas mejoras para un trabajo futuro.

# 2. Estado del Arte

## 2.1. Introducción

En este capítulo se presentan diferentes sensores que pueden ser usados para medir vibraciones, así como las tecnologías que pueden ser usadas para detectar defectos. Además se presentan diferentes trabajos relacionados con el proyecto, en los que se ha usado un acelerómetro para detectar las vibraciones en las vías ferroviarias, y así determinar defectos en estas. La decisión sobre las tecnologías usadas para la resolución del problema se explican en el apartado de diseño.

## 2.2. Sensores usados para medir vibraciones

### 2.2.1. Acelerómetro

Un acelerómetro [2] es un dispositivo capaz de medir la aceleración de un cuerpo, lo que implica que es capaz de medir vibraciones. Hay varios tipos de acelerómetros, siendo los más importantes el mecánico y el piezoeléctrico.

Para la detección de vibraciones, los dispositivos más usados son los acelerómetros piezoeléctricos. Estos acelerómetros generan una carga eléctrica proporcional a la fuerza cuando se produce una vibración, debido a que la masa (mostrada en la Fig. 1) comprime el material piezoeléctrico [2].

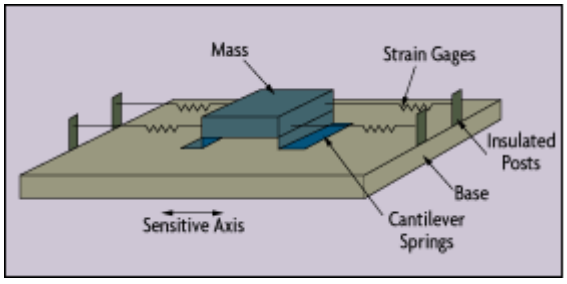


*Fig. 1. Esquema de un acelerómetro (Figura tomada de [2])*

Dado que el acelerómetro mide cambios en la aceleración de un cuerpo, es un instrumento muy común en la medida de vibraciones. Sin embargo, existen actualmente acelerómetros de tres ejes, que realizan a nivel microscópico los procesos anteriormente mencionados y son capaces de medir aceleraciones en los ejes X, Y, Z.

### 2.2.2. Giroscopio

Un giroscopio es un instrumento encargado de medir las velocidades angulares. Puede ser analógico o electrónico, pero el que concierne a este proyecto solo es el electrónico. Por lo tanto, de ahora en adelante cuando se hable de giroscopio se refiere a un sensor electrónico capaz de medir la velocidad angular.

Básicamente, un giroscopio mide rotaciones. Si se integra la velocidad angular con respecto del tiempo se obtiene el desplazamiento angular, por lo que también es una opción viable para medir vibraciones.

En la actualidad tanto los acelerómetros como los giroscopios son de suma importancia en los teléfonos móviles, que suelen tenerlos integrados. Gracias al acelerómetro rota la pantalla del móvil cuando cambia la orientación del mismo, debido a la aceleración que se produce al realizar este movimiento, y gracias al giroscopio se pueden ver videos 360º, en los que los movimientos a realizar son más suaves y se calculan las velocidades angulares que permiten seguir el movimiento realizado por el usuario para visualizar el video [3].

### 2.2.3. IMU

Una IMU [4] o una unidad de medición inercial, es un dispositivo electrónico que generalmente combina uno o varios acelerómetros y giroscopios, con el fin de medir velocidad y rotación al mismo tiempo. Algunas veces se pueden usar también magnetómetros, obteniendo medidas más precisas (generalmente para localización).

En definitiva, es una opción mucho más completa que las anteriores en cuanto a medir vibraciones se refiere, pues al constar de un acelerómetro y un giroscopio que trabajan en conjunto, puede obtener medidas más reales que uno de los sensores trabajando de manera individual, sobre todo en movimientos en los que el acelerómetro no es capaz de detectar movimiento y el giroscopio sí, y viceversa. En la Fig. 2 se muestran los ejes de trabajo de una IMU de 9 DOF (Degrees of Freedom) o grados de libertad. Es decir, los tres ejes de medida del acelerómetro, del magnetómetro y del giroscopio.


*Fig. 2. Ejes de trabajo de una IMU de 9 DOF (Figura tomada de [4])*

## 2.3. Metodologías existentes para la detección de defectos

En este apartado se mencionarán las metodologías que podrían usarse para la detección de defectos en los raíles ferroviarios. Todas ellas son END (Ensayos No Destructivos), pues las vías deben de seguir siendo útiles después de realizar la inspección.

### 2.3.1 Radiografía

La radiografía puede detectar defectos internos de un material sólido debido a la absorción (en mayor o menor medida) de radiaciones ionizantes cuando atraviesa el material, lo que produce una imagen visual del defecto [5]. Cuanto mayor sea la profundidad de la pieza, mayor energía de radiación se necesita. En la actualidad se pueden usar rayos X o rayos Gamma. En la Fig. 3 se detallan algunas diferencias en las características de ambos:


*Fig. 3: Diferencias entre rayos X y rayos Gamma (Figura tomada de [5])*

Una radiografía es una buena opción pues puede usarse tanto en materiales ferromagnéticos como aquellos que no lo son, y se puede detectar de forma fácil el defecto. Sin embargo, es difícil de usar en piezas de geometría compleja y no se trata de una tecnología barata.

### 2.3.2 Ultrasonidos

El ensayo de ultrasonido es un método de ensayo no destructivo (END). Utiliza ondas de sonido de alta frecuencia para localizar y medir las discontinuidades internas de una pieza [5]. Se basa en un transductor que emite una señal acústica y produce un fenómeno de reflexión en el defecto permitiendo determinar la distancia al defecto,

mediante el tiempo transcurrido en reflejar la onda, y la longitud de este, mediante la amplitud de la onda.

La onda reflejada, llega al transductor y es convertida en un impulso eléctrico, que permite ver dónde se encuentra y cómo de grande es el defecto [6]. En la Fig. 4 se refleja el funcionamiento de los ultrasonidos.

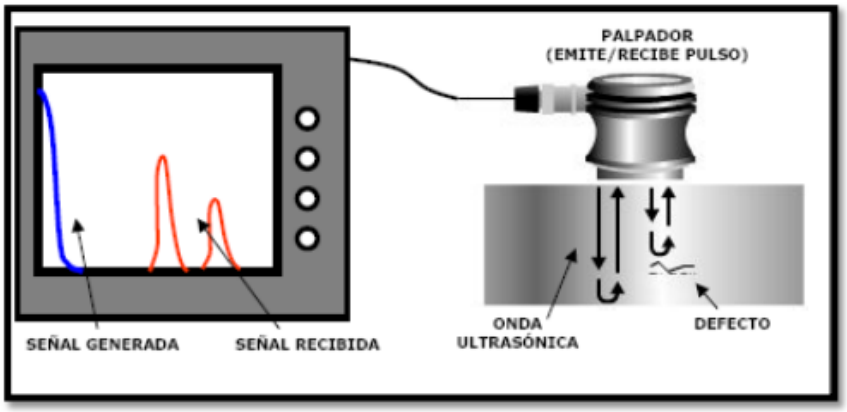


*Fig. 4. Funcionamiento de ensayo por ultrasonidos (Figura tomada de [5])*

Al igual que las radiografías, los ultrasonidos se pueden emplear tanto en materiales ferromagnéticos como en no ferromagnéticos, pero con el inconveniente de que no es capaz de detectar defectos superficiales, y que no es muy efectivo en piezas con alta porosidad, pues se podrían obtener resultados erróneos.

### 2.3.3. Process Compensated Resonance Testing (PCRT)

El PCRT [7], o en español, Inspección por Resonancia con Compensación, se trata de un NDT (Non-Destructive Testing) o END (Ensayo No Destructivo). Su funcionamiento está basado en la medida de frecuencias de resonancia, que están determinadas por las propiedades elásticas, de geometría y masa de los objetos. La presencia de un defecto estructural cambia la rigidez del objeto, provocando un cambio en las frecuencias de resonancia. Por lo que, debido a esto, una vibración puede predecir la integridad estructural de una pieza.

El PCRT se puede realizar en piezas cerámicas y metálicas. Encuentra cambios estructurales mediante el empleo de resonancias mecánicas, de forma que se pueden detectar piezas que tienen defectos e incluso detectar el comienzo del fenómeno de fatiga antes de que aparezca el defecto [8], a diferencia de otros ensayos no destructivos como los mencionados anteriormente, que simplemente encuentran las imperfecciones ya existentes.

Los costes de este ensayo son elevados, pero si el tamaño de la serie de piezas a inspeccionar es grande, estos costes son justificables, pues los tiempos de inspección son mucho menores que los relativos a otro tipo de ensayos no destructivos.

Aunque este tipo de ensayo no sea una opción viable a emplear en la inspección de las vías ferroviarias se ha mencionado en esta memoria pues utiliza las vibraciones para la detección de defectos, lo mismo que se pretende hacer en la ejecución de este proyecto.

## 2.3. Trabajos previos

### 2.3.1. Proyecto fin de carrera: M. González, Universidad Carlos III de Madrid

En el año 2012, Miguel González realizó su proyecto de fin de carrera en la Universidad Carlos III de Madrid [9], titulado "Registro y análisis de la señal vibratoria de un eje ferroviario a escala para defectología". En él, se analiza la vibración de dos ejes ferroviarios, mediante diversos procedimientos y usando un acelerómetro para la detección de vibraciones.

En este proyecto, se presenta un mantenimiento predictivo de las máquinas, basándose en las vibraciones, y se caracterizan los defectos según las señales vibratorias. También se realiza un estudio más amplio de la onda de vibración, que es necesario para la caracterización del defecto según la vibración que produce la máquina.

De forma similar que en el proyecto a realizar, se hace uso de un acelerómetro para medir las vibraciones de la maquinaria. Sin embargo, el presente proyecto se va a centrar en la detección de irregularidades en la superficie de las vías ferroviarias, a diferencia de Miguel González que analiza en su proyecto diversos problemas, como el desalineamiento de la máquina con el eje o un desequilibrio del acoplamiento.

### 2.3.2. FerroDron

FerroDron es el proyecto base a partir del cual surge el presente Trabajo de Fin de Grado. Se trata de un proyecto llevado a cabo por la Universidad Carlos III de Madrid [10], en conjunto con la empresa TECSA, líder especialista en vías del país. El proyecto fue financiado por el Ministerio de Economía y Competitividad de España [11].

Con FerroDron se propone la creación de "un vehículo ultraligero no tripulado (véase Fig. 5) desplazable sobre carriles para labores automáticas de inspección, mantenimiento y vigilancia de vía e infraestructura ferroviaria con posicionamiento y comportamiento inteligente mediante visión por computador y sensores combinados" [12]. En definitiva, se trata de un dron ferroviario automatizado que realiza labores de inspección y mantenimiento de las vías, enviando datos de forma automática a un puesto de control en tiempo real.

Algunos de las funciones que FerroDron puede hacer es por ejemplo la construcción de una imagen virtual del recorrido que realiza gracias a los sensores que tiene incorporados, como la visión artificial.

*Fig. 5. FerroDron (Figura obtenida de [12])*

La idea del proyecto es que el sensor utilizado en el trabajo esté colocado en este vehículo que trabajando de manera conjunta con el resto de los sensores integrados en el mismo, proporcione datos exactos sobre el estado de los raíles, dado que en la actualidad solo se captura el entorno ajeno a las vías

Mencionar que el presente proyecto no es una ampliación de FerroDron, sino la idea teórica de que el sensor inercial, debido a los requisitos buscados (autonomía y automatización) funcionaría en concordancia en este vehículo, pues su función es la inspección y mejora del mantenimiento de las vías ferroviarias. Además, como se explicará posteriormente, en este proyecto se hará uso de ROS para que el sensor escogido publique datos en conjunto con el resto de los sensores y se pueda proporcionar datos precisos sobre el estado de las vías.

# 3. Diseño

En el presente capítulo se aborda el motivo de elección del sensor, controlador, lenguaje y entorno de programación. En definitiva, se justifican todas las elecciones de diseño para la realización del proyecto.

## 3.1. Elección del sensor

En primer lugar, y con lo expuesto anteriormente sobre los diferentes sensores que hay en el mercado que se pueden utilizar para medir vibraciones en la actualidad, se ha elegido un IMU o sensor inercial para la realización del proyecto.

El motivo principal es que es capaz de realizar medidas más precisas y dado que el sensor irá colocado en los soportes de las ruedas del vehículo, el trabajo conjunto del acelerómetro y el giroscopio permitirán una mayor exactitud a la hora de detectar imperfecciones en las vías ferroviarias. Teniendo esto claro, y entre los cientos de sensores inerciales que hay en el mercado, se ha elegido el MPU-6050 de Invensense.

El MPU-6050 [13] es un sensor inercial de 6 DOF, que combina un giroscopio de 3 ejes y un acelerómetro de 3 ejes (6 grados de libertad). Trabaja con un voltaje entre 3 y 5 voltios. Además, tiene incorporado un Digital Motion Processor (DMP) o procesador digital de movimiento, con un puerto I2C auxiliar para poder conectar a dispositivos externos, como un magnetómetro, y proporcionar una salida de 9 DOF. La comunicación con el controlador se realiza mediante el bus I2C. Posteriormente se hará una explicación sobre este modo de comunicación.

Por lo tanto, el MPU-6050 (Fig. 6) es una opción completa para el proyecto a realizar, siempre y cuando se utilice un microcontrolador apropiado para poder establecer la comunicación I2C.

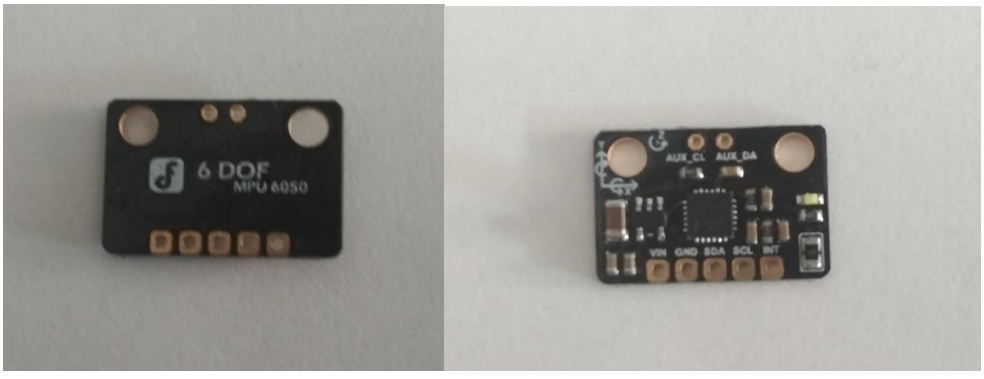


*Fig. 6. Sensor MPU-6050*

## 3.2. Elección del microcontrolador

A continuación se exponen las posibles opciones de microcontroladores para el desarrollo del proyecto, y las principales diferencias entre ellos.

Los microcontroladores que se han tenido en cuenta para el desarrollo de este proyecto han sido: Arduino, Parallax Basic Stamp 1 y BX-24 de Netmedia.

- **Placa Arduino UNO:** Arduino UNO [14] se trata de una placa microcontroladora desarrollada por Arduino. Puede adquirir entradas y convertirlas en salidas (analógicas o digitales). Arduino es una plataforma de código abierto, por lo que es muy fácil encontrar código útil para los proyectos y reutilizarlo. La placa Arduino UNO posee 14 pins digitales de entrada / salida y 6 entradas analógicas, además de conexiones de alimentación y tierra [15]. Es una opción barata y sencilla de usar, pues su programación, en lenguaje Arduino, es muy intuitiva.
- **Parallax Basic Stamp 1:** se trata de un microcontrolador desarrollado por Parallax Inc. Posee 8 pins de entrada / salida [16] y su programación es en el lenguaje PBASIC 1. Es una opción básica y económica, ideal para proyectos que no requieran de una alta velocidad de procesamiento. El microcontrolador se muestra en la Fig. 7.

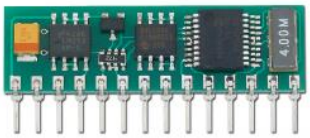

*Fig. 7. Microcontrolador Basic Stamp 1 (Figura obtenida de [16])*

- **BX-24:** este microcontrolador es fabricado por Netmedia. Según su página web, es "el microcontrolador programable BASIC más potente del mundo" [17]. Posee 16 pins de entradas/salidas (8 digitales y 8 analógicas), y puede ejecutar 83000 instrucciones por segundo. En la Fig. 8 se puede ver dicho microcontrolador.

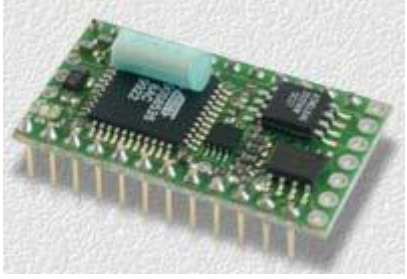

*Fig. 8. Microcontrolador BX-24 (Figura obtenida de [17])*

De las opciones anteriores, la opción elegida ha sido la placa Arduino UNO (Fig. 9). Aunque el BX-24 pueda parecer la mejor opción, y la Basic Stamp la más económica, Arduino es la opción más completa. Se ha escogido una placa Arduino UNO para la realización del proyecto principalmente por la simplicidad de uso. Con la placa Arduino, es más fácil la conectividad con el sensor, además de la existencia de numerosos manuales para realizar la conexión entre el MPU-6050 y Arduino, y obtener los datos que este mide de una forma sencilla. Además, el autor del proyecto ya se encontraba en posesión de una placa Arduino UNO. Otro de los motivos de la elección de Arduino es la gran interacción con ROS, como se comentará de manera posterior.

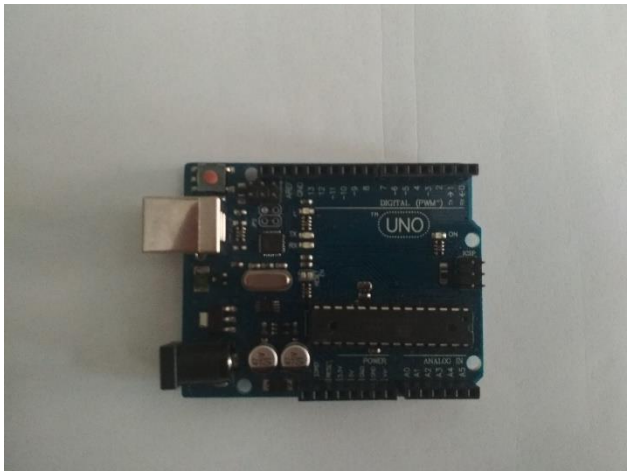


*Fig. 9. Placa Arduino UNO*

Por otro lado, la programación en Arduino es intuitiva y al ser una plataforma de código abierto, hay multitud de librerías de código reutilizable que permite la conexión I2C, de la que se requieren conocimientos sobre el campo de las comunicaciones para poder llevar a cabo.

El I2C bus (Inter-Integrated Circuit) [18] o Circuito Inter-Integrado es un modo de comunicación para realizar una transmisión bidireccional de datos entre distintos microcontroladores. Aunque es mucho más complejo, básicamente se usan dos líneas: el SDA para transmitir datos, y el SCL que es el reloj serie. El SCL se usa para sincronizar los datos del SDA. En I2C siempre hay maestros y esclavos. El maestro es el que maneja la línea de reloj SCL. En el caso de este proyecto, el maestro sería la placa Arduino y el esclavo el sensor inercial MPU-6050.

Con lo dicho, en Arduino se pueden encontrar tres librerías esenciales para la correcta lectura de datos del sensor:

- **“Wire.h”:** permite comunicar con dispositivos I2C [19]. La línea SDA irá conectada al pin A4, y la SCL al pin A5.
- **“I2Cdev.h”:** librería complementaria a *Wire.h* para una conexión más simple con dispositivos I2C.

- **"MPU6050.h":** sirve para simplificar la programación de la adquisición de datos del sensor MPU-6050.

En el apartado de implementación se comentará el código de programación para la conexión y obtención de datos del sensor.

## 3.3. Filtro para los datos

Un gran inconveniente de obtener medidas del sensor inercial es que en las mediciones realizadas se obtiene mucho ruido. Esto significa que los datos obtenidos pueden tener un cierto margen de error. Para obtener mediciones con menor error sería necesario aplicar un filtro reduzca el error y nos proporcione datos más exactos. En la Fig. 10 se puede apreciar en rojo una serie de valores sin filtrar, y en azul esos mismos valores filtrados.

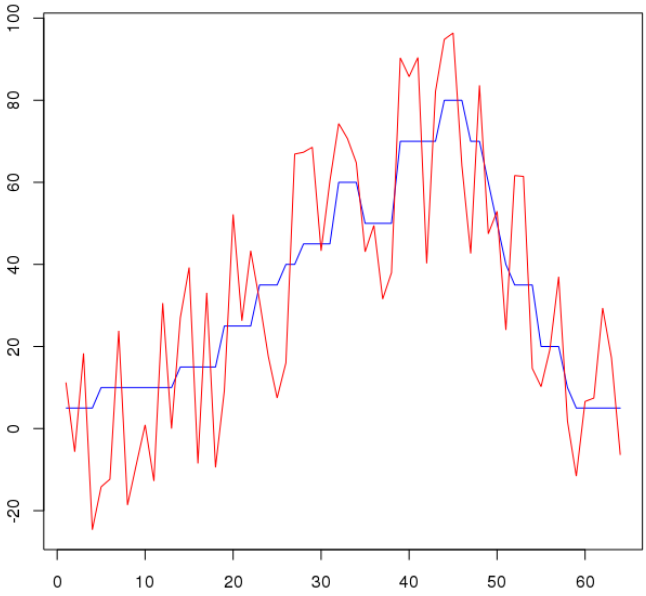


*Fig. 10. Datos filtrados y sin filtrar (Figura obtenida de [20])*

Uno de los filtros más importantes y más conocidos es el **filtro de Kalman** [21]. Desarrollado por Rudolf Kalman en los años 60, es capaz de predecir una serie de datos sin error a partir de unas medidas que pueden obtener error, mediante estimaciones y una serie de algoritmos. A pesar de ser una buena opción por la precisión de los valores estimados, precisa de cálculos complejos, lo que implica un tiempo de procesamiento elevado.

Otro filtro conocido es el **filtro complementario**. Según la entrada de Luis Llamas en su blog [22], "el filtro complementario puede considerarse una simplificación del filtro de Kalman que prescinde del análisis estadístico". Es decir, con el filtro complementario se pueden obtener medidas filtradas y por lo tanto sin ruido, no tan exactas como las del filtro de Kalman, pero sí bastante precisas. Sin embargo, se trata

de un filtro que se caracteriza por la sencillez a la hora de ser implementado, y que a diferencia del filtro de Kalman, no precisa de tiempos de procesamiento tan elevados.

Si se implementase algún filtro en el proyecto el elegido sería el filtro complementario, por lo comentado anteriormente. Pero como se verá en el apartado de resultados el proyecto no ha llegado a realizarse (no se ha podido colocar el sensor inercial en FerroDron), y por lo tanto no han podido obtenerse datos reales de la vía. Pero en el caso de que el proyecto se llegase a realizar, sí sería necesario aplicar un filtro para eliminar el ruido y obtener así los datos más exactos posibles.

## 3.3. Recolección de datos

Una vez iniciado el proyecto, se vio la necesidad de recolectar los datos recogidos por el sensor para poder ser analizados y representados gráficamente en la presente memoria. Para ello, de manera inicial se pensó en realizar una conexión entre Arduino y Excel para que los datos recogidos por el sensor inercial fueran publicados directamente en una hoja de Excel. Sin embargo, esto implicaba un nivel de conocimiento en Excel sobre macros, programadas en lenguaje VBA (Visual Basic for Applications) desconocido por el autor, que requería invertir más tiempo en su aprendizaje que en la realización del proyecto en sí mismo.

Una vez rechazada esta propuesta, y dado que la necesidad de recoger los datos es solo para la memoria y no para la realización del proyecto en sí (pues estos datos son publicados con ROS), se decidió exportar los datos a una tarjeta microSD para luego poder analizarlos con Microsoft Excel.

Para ello era necesario un módulo lector microSD adaptado para Arduino. El dispositivo elegido fue el módulo lector microSD de Catalex, distribuido por kwmobile [23]. Precisa de una alimentación de 5V, y soporta tarjetas SDHC de hasta 32 GB, que fue la tarjeta usada. Este adaptador se muestra en la Fig. 11.

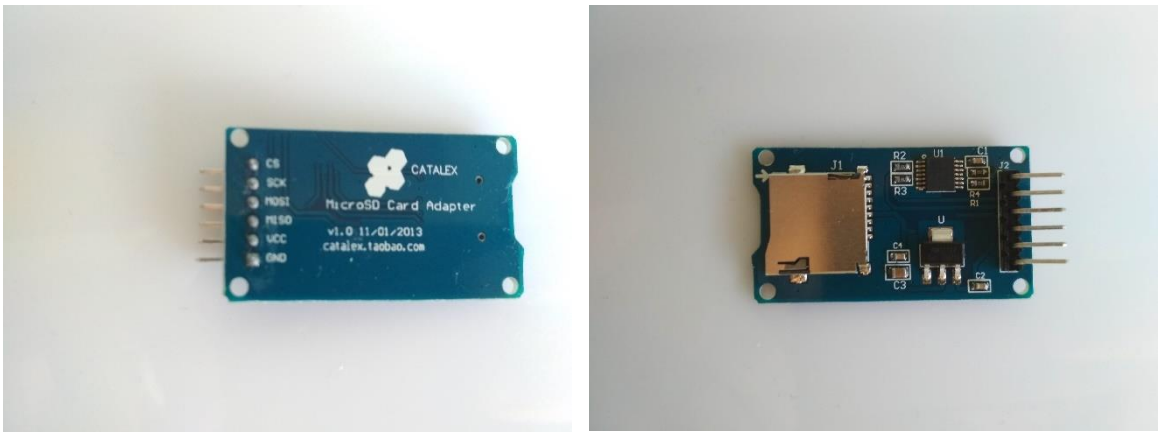


*Fig. 11. Módulo lector de tarjeta microSD Catalex*

El procedimiento que se sigue es el siguiente:

1. **Implementación de código:** se implementa el código necesario para la creación de un archivo de texto que recoja los datos obtenidos por el sensor en la tarjeta microSD.
2. **Importación de datos en Microsoft Excel:** se importan los datos en Excel a partir archivo de texto de la aceleración y la velocidad angular.
3. **Visualización de los datos:** una vez en Excel, los datos pueden ser tratados con más facilidad y pueden ser visualizados en forma de gráfica, que permite ver de forma clara las vibraciones y cuando hay un defecto.

La conexión del adaptador y la implementación del código serán comentados posteriormente en el apartado de Implementación.

## 3.4. Integración con ROS

### 3.4.1. Introducción

ROS (Robot Operating System) [24] se trata de un entorno de trabajo flexible, basado en una colección de herramientas, librerías y paquetes que simplifican la tarea de crear software para robots. ROS, al igual que Arduino es un proyecto "open source" o de código abierto, que cuenta con una amplia comunidad de miembros, por lo que se mantiene constantemente actualizado y es una comunidad muy activa.

ROS se instala sobre un sistema operativo (generalmente Linux), por lo que puede ser considerado un metasistema operativo.

### 3.4.2. Conceptos de ROS

En este apartado se presentan los conceptos básicos para el entendimiento de ROS.

- **Master:** el ROS Master [25] es la base de ROS. El papel del Maestro es permitir que los nodos de ROS puedan comunicarse entre sí, mediante el intercambio de mensajes. Para que sea ejecutado, se debe introducir el comando *roscore* en la consola de comandos.
- **Nodos:** un nodo [26] es un proceso que realiza la computación. Un sistema de control de un robot estará compuesto de varios nodos. Por ejemplo, un nodo controla el motor de las ruedas de un robot, otro el movimiento de los brazos, etc. El uso de nodos proporciona una mayor facilidad en la detección de fallas en el código, ya que cada nodo se encarga de una cosa distinta. Los nodos se pueden comunicar entre sí publicando y/o suscribiéndose a un topic.
- **Mensajes:** los nodos se comunican unos con otros publicando mensajes en los topics [27]. Un mensaje es una estructura de datos simple, y pueden ser primitivos como los tipo enteros, booleanos, caracteres, etc; o arrays de los anteriores. Los mensajes también pueden ser estructuras, como las estructuras de C.

- **Topics:** los nodos se comunican unos con otros intercambiando mensajes a través de los topics [28]. Los nodos pueden publicar en uno o varios topics, o suscribirse a ellos. Así mismo, puede haber múltiples publicadores y suscriptores a un topic. En cualquier caso, el transporte de información es en un único sentido.

Con estos conceptos fundamentales sobre ROS se pueden tener unas nociones básicas para cuando se use lenguaje como "nodo" o "topic" en el proyecto.

### 3.4.3. Por qué ROS

Como se ha visto, ROS es una opción muy completa a la hora de realizar proyectos de robótica. Al ser una plataforma de código abierto hay multitud de código reutilizable que permite realizar proyectos de robótica completos. Además, cada sensor o motor de un robot puede publicar o recibir datos independientemente en diferentes topics permitiendo que cada uno trabaje de manera autónoma, ya que los nodos no se comunican directamente unos con otros, sino que lo hacen a través de los topics.

Dado que la idea es integrar el sensor inercial en el FerroDron, es necesario que dicho sensor publique sus datos con ROS al igual que hacen el resto de sensores para que el software pueda combinarlos y analizarlos, determinando así el estado de las vías ferroviarias. De esta manera se justifica la elección de ROS para la realización del proyecto.

### 3.4.4. rosserial_arduino

rosserial_arduino [29] es un paquete que contiene extensiones específicas de Arduino, necesarias para ejecutar rosserial_client en un Arduino. Es decir, una vez con ROS instalado en el sistema, este paquete permite integrar de manera relativamente sencilla ROS en Arduino mediante la creación de nodos y topics directamente desde el código ideado en el software Arduino IDE.

Para el proyecto se ha elegido esta opción para la integración con ROS en lugar de la programación directamente en el mismo debido a la simplicidad con la que se pueden crear nodos, topics, e intercambiar mensajes programando en el mismo código realizado para Arduino.

Por otro lado, las medidas del sensor pasan directamente al nodo que las publica en un topic sin perder información en el proceso, como se explicará con posterioridad en el apartado de implementación. Por lo tanto, se ha implementado el código para configurar el MPU-6050 y que adquiera datos, y en base a eso se ha ideado el código para crear el nodo que publicará en los distintos topics las medidas obtenidas.

## 3.5. Elección del sistema operativo

A priori se escogió Windows para la realización del proyecto, dada la simplicidad de uso, la experiencia del autor en este sistema operativo y su compatibilidad con la

plataforma Arduino IDE. Pero una vez se empezó a trabajar con ROS se vio claro que la elección lógica era Linux, pues, aunque existen paquetes para poder trabajar con ROS en Windows, era mucho más intuitivo la instalación y el trabajo en Linux. Además, la plataforma Arduino IDE se podía usar en este sistema operativo y permitía trabajar con el paquete rosserial_arduino, lo que facilitaba la creación de código.

En cuanto a la creación de la memoria y a la representación gráfica de los datos se han realizado en el sistema operativo Windows.

# 4. Implementación

En este capítulo se procederá a la explicación de las conexiones entre el sensor y Arduino, del código creado y de cómo se ha realizado la integración del sensor con ROS. Por otra parte, se ejecutará una prueba para comprobar que tanto el código como las conexiones han sido realizadas correctamente.

## 4.1. Medios utilizados

A continuación se detallan los equipos y programas usados para la realización del presente Trabajo de Fin de Grado.

**Equipos:**

- Ordenador portátil Toshiba Satellite, finales de 2014.
  - Procesador Intel Core i7, 3.1 GHz
  - Memoria 8GB 1600 MHz DDR3L
  - Gráficos AMD Radeon R7 M260 2GB
  - Disco duro de 1 TB
  - Cuenta con partición para cambiar el sistema operativo.
  - Sistema Operativo principal: Windows 10
  - Sistema Operativo secundario (Linux): Ubuntu 16.04

**Programas:**

- Arduino IDE 1.8.6 para Linux: entorno de desarrollo del código necesario para el sensor inercial y su sincronización con la placa Arduino.
- ROS Kinetic: no es un programa como tal, pero es la plataforma en la que el sensor publica los datos recogidos, como se ha explicado anteriormente.
- Fritzing 0.9.3b.32 para Windows: programa que permite realizar esquemas de conexión de la placa Arduino con una amplia variedad de componentes electrónicos.
- Microsoft Word 2016 para Windows: editor de texto utilizado para la realización de la memoria del proyecto.
- Microsoft Excel 2016 para Windows: aplicación de hojas de cálculo utilizada para analizar los datos recogidos por el sensor y representarlos gráficamente.

## 4.1. Conexión del sensor inercial

El primer paso en la realización del trabajo es conectarlo a la placa Arduino UNO. Para ello, se ha tenido en cuenta que la comunicación entre el sensor y la placa es I2C. Como se ha mostrado en el apartado anterior, con la librería *Wire.h* de Arduino se podía realizar este tipo de comunicación.

La línea SDA irá conectado al pin A4 de la placa, y la línea SCL al pin A5. Además, era necesaria la conexión a tierra y la alimentación del sensor. En este caso, el GND (tierra) del sensor es conectado al GND del Arduino, y la entrada Vin (alimentación) del sensor a la salida de 3.3V que proporciona la placa. También podría haber sido conectada a la salida de 5V, pues el rango de funcionamiento del sensor está entre esos dos valores. En la Fig. 11 se muestra un esquema de la conexión a realizar.

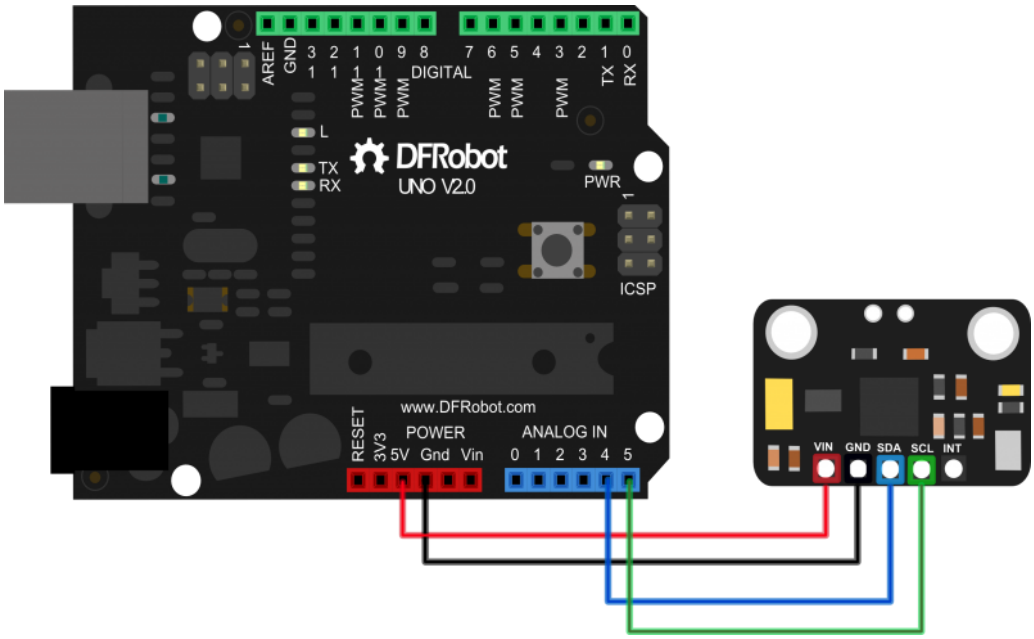


*Fig. 12. Esquema de las conexiones a realizar (Figura obtenida de [13])*

## 4.2. Conexión del módulo lector de microSD

Como ya se ha comentado anteriormente, aunque en la realización del proyecto no es necesario, sí que se necesitaba una manera de recoger los datos del sensor para ser analizados y comentados. El camino elegido para adquirir estos datos en un archivo de texto fue la exportación de estos a una tarjeta microSD.

Para conectar el módulo adaptador de microSD con Arduino se ha utilizado la comunicación SPI y la librería "SD.h" [30]. La comunicación SPI (Serial Peripheral Interface) [31] tiene básicamente tres líneas normalmente comunes a todos los dispositivos que utilizan este tipo de comunicación:

- **MISO (Master In Slave Out):** línea de comunicación del esclavo al maestro.
- **MOSI (Master Out Slave In):** línea de comunicación del maestro al esclavo.
- **SCK (Serial Clock):** señal de reloj que sincroniza la transmisión de datos generada por el maestro.

Además se suele tener otra línea adicional, **SS (Slave Select)** para seleccionar el dispositivo con el que se quiere establecer comunicación.

La línea MOSI irá conectada al pin D11, MISO al D12 y SCK al D13. CS (para seleccionar el dispositivo) se conecta al pin D9, y Vcc y GND se conectan respectivamente a la salida de 5V de la placa y a una toma de tierra (GND) de la misma. El esquema de conexiones a realizar se muestra en la Fig. 12, y en la Fig. 13 se puede encontrar las conexiones completas tanto del sensor como del módulo lector de microSD.

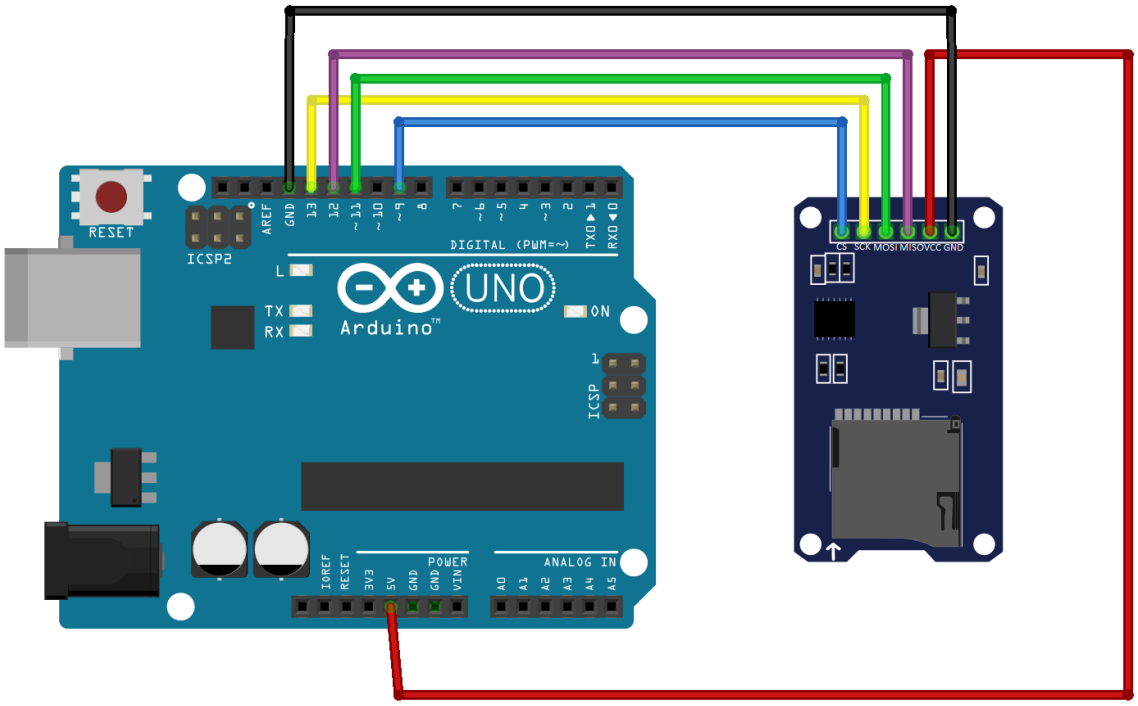


*Fig. 13. Esquema de conexión con módulo lector microSD*

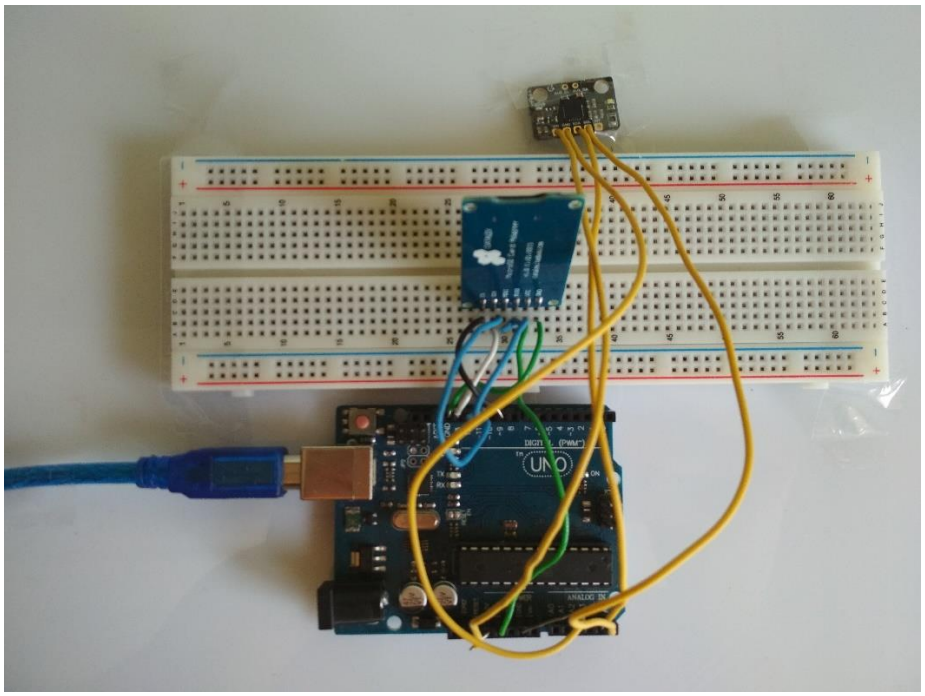

*Fig. 14. Conexión completa del sensor y del módulo adaptador*

## 4.2. Lectura de valores del sensor

Antes de la integración con ROS, era necesario comprobar si las conexiones estaban bien realizadas y que el sensor funcionaba correctamente. El sensor inercial debía detectar vibraciones y era necesario cargar un código en la placa Arduino para que realizase dicha tarea. El código base se obtuvo del manual dispuesto por el fabricante DFRobot [13], donde también se indicaban las conexiones que se debían realizar.

### 4.2.1. Programación en Arduino

Para empezar, se hace una breve explicación de la forma de programar en Arduino.

Un proyecto de Arduino se denomina *sketch* y tiene la extensión *.ino*. Como mínimo, un sketch se compone de dos funciones principales:

- ***void setup():*** esta función se convoca una sola vez, cuando el programa empieza. En ella se realizan las configuraciones oportunas para el correcto funcionamiento del programa, como la inicialización de librerías o la configuración de un pin (si es de salida o de entrada).
- ***void loop():*** es llamada repetidamente y es dónde se realiza el funcionamiento del programa.

Ambas funciones tienen que ser contenidas en el sketch para el correcto funcionamiento del programa.

### 4.2.2. Código para la lectura de valores iniciales

Se procede a la explicación del código inicial obtenido de DFRobot [13], que se adjunta en el Anexo 1. A este código se le ha añadido lo necesario para exportar los datos obtenidos a la tarjeta microSD, basándose en el código creado por Luis Llamas en su blog [32].

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include <SD.h>
```

En estas líneas, se procede a la llamada de las bibliotecas necesarias para la conexión con el sensor y la lectura de datos, cuya función se ha descrito anteriormente. Además se incluye la biblioteca “SD.h” que permite la conexión con el módulo adaptador que se ha comentado antes.

```
File Datos;
```

Se inicializa el archivo de texto que se va a escribir.

```
MPU6050 accelgyro;
```

Crea una clase tipo MPU6050 llamada “accelgyro”.

```
int16_t ax, ay, az;
int16_t gx, gy, gz;
```

Se definen las variables para la lectura de las aceleraciones y de las velocidades angulares en los ejes x, y, z.

```
#define LED_PIN 13
bool blinkState = false;
```

Se establece que el led del Arduino está en el pin 13 y se define una variable booleana para mostrar la actividad del sensor. “False” apagará el led, “true” lo encenderá.

La siguiente parte del código se encuentra dentro de la función setup():

```
Wire.begin();
Serial.begin(38400);
```

```
Serial.println("Initializing I2C devices...");
accelgyro.initialize();
```

Se inicia la conexión con el sensor, se establece que los valores que se reciben por el puerto serial sean a una velocidad de 38400 baudios, y se inicializa el sensor. Al inicializar el sensor, los rangos por defecto si no se especifican son:

- Acelerómetro: ±2 g (fuerza de gravedad)
- Giroscopio: ±250 º/s (grados/segundos)

La función *Serial.println()* imprime por pantalla el texto indicado con un salto de línea posterior, a diferencia de la función *Serial.print()*, que no provoca cambio de línea tras la impresión.

```
pinMode(LED_PIN, OUTPUT);
```

Se configura el pin 13 (led del Arduino) de salida.

```
Serial.print("Iniciando tarjeta microSD:");
if (!SD.begin(9)) {
  Serial.println("No se ha podido iniciar");
}
Serial.println("Iniciado correctamente");
```

Se inicia la tarjeta SD. Si no está conectada al pin D9 (CS), muestra un mensaje de error.

La siguiente parte del código se encuentra dentro de la función loop():

```
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

Obtiene los valores sin procesar (en bruto) de las aceleraciones y las velocidades angulares en sus tres ejes.

```
Serial.print("a/g:\t");
Serial.print(ax);
Serial.print("\t");
Serial.print(gx);
Serial.print("\t");
```

Se imprime por pantalla los valores de las aceleraciones y las velocidades angulares.

```
Datos = SD.open("valoresiniciales.txt", FILE_WRITE);
```

Se abre el archivo de texto “valoresiniciales.txt”. Si no existe, se crea uno. El archivo será de escritura (FILE_WRITE).

```
if (Datos) {
        Datos.print("a/g:\t");
        Datos.print(ax);
        Datos.print("\t");
        Datos.print(gx);
        Datos.print("\t");

        Datos.close();
  }
  else {
    Serial.println("Error al abrir el archivo");
  }
  delay(10);
```

Si el archivo de datos existe y no está dañado se escriben los mismos, y si no se manda un mensaje de error. Se introduce un retardo de 10 milisegundos para prevenir que el programa se bloquee.

```
blinkState = !blinkState;
digitalWrite(LED_PIN, blinkState);
```

Hace parpadear el led del Arduino mientras se recogen los datos del sensor.

Con este código inicial se obtienen datos sin tratar del sensor que se mostrarán en el apartado 5 de la memoria, correspondiente a los resultados.

### 4.2.3. Código para la lectura de valores escalados

Para poder obtener valores entendibles se aplicaron ciertos cálculos que se han obtenido del manual de especificaciones de producto [33], para que los valores de aceleración se obtuviesen en $m/s^2$ y los valores de velocidad angular en $^o/s$, pues los datos obtenidos en el apartado anterior son ininteligibles.

El código es similar al de las lecturas iniciales, pero haciendo las conversiones de unidades en él. El código se adjunta en el Anexo 2, pero a continuación se comentan dichos cambios. Los cambios se realizan dentro de la función *loop()*, posteriormente de

que se hayan leído los datos del sensor con la función *getMotion6()*. Para los valores del acelerómetro:

```
float ax_m_s2 = ax * (9.81/16384);
```

Se crea una variable de tipo float que será igual al valor obtenido inicial, cuya unidad es "g" o gravedad. 1 fuerza de gravedad es igual a 9,81 $m/s^2$. Se divide por 16384 para escalarlo a valores de aceleración, puesto el que el sensor está inicializado en ±2 g, y así lo indica en las especificaciones de producto. Lo mismo se haría para las aceleraciones en los ejes Z e Y.

Para los valores del giroscopio:

```
float gx_r = gx / 131;
```

Se crea una variable de tipo float que será igual al valor obtenido inicial, cuya unidad es "º/s". Para escalar el valor a unidades de velocidad angular, se divide por 131 porque así lo indican las especificaciones cuando el sensor esta inicializado a ±250 º/s.

Una vez escalados, los datos obtenidos son entendibles y se pueden tratar con mayor facilidad. Se muestran en el apartado de resultados.

## 4.3. Implementación de ROS

Como se ha comentado anteriormente, se ha usado el paquete "rosserial_arduino" para la publicación de los datos obtenidos por el sensor en ROS. Para ello se ha modificado el código anterior introduciendo lo necesario para la creación de los nodos, topics y mensajes correspondientes directamente en Arduino. Comentar que para este apartado que es el de la ejecución real del proyecto se ha excluido el módulo adaptador de microSD así como el código creado anteriormente pues los datos son directamente publicados en ROS.

El código completo se incluye en el Anexo 3, aunque en este apartado se procede a la explicación de las líneas más significativas del mismo. El código se ha creado siguiendo el ejemplo que ROS ha publicado en su página de cómo crear un *"Publisher"* [33].

```
#include <ros.h>
```

Necesario para poder implementar un programa en ROS.

```
#include <std_msgs/Float64.h>
```

Se incluye la librería del tipo de mensajes que se quiere que los nodos publiquen en los topics. En el caso de este proyecto, los datos una vez transformados son de tipo float.

```
ros::NodeHandle nh;
```

Se instancia el "node handle", que permitirá crear "publishers" y "subscribers" (publicadores y suscriptores).

```
std_msgs::Float64 ax_msg, ay_msg, az_msg, gx_msg, gy_msg, gz_msg;
```

Se crean los mensajes para las distintas variables del acelerómetro y el giroscopio.

```
ros::Publisher ax_pub("axx", &ax_msg);
```

Se crean los 6 topics (3 del acelerómetro y 3 del giroscopio).

```
nh.initNode();
nh.advertise(ax_pub);
```

Dentro de la función *setup()* es necesario inicializar el "node handle" y "avisar" al topic en el que se va a publicar.

```
ax_msg.data = ax_m_s2;
```

Dentro de la función *loop(), l*os mensajes a publicar en el topic serán los datos recogidos por el sensor.

```
ax_pub.publish( &ax_msg );
```

Se publica en el topic el mensaje

```
 nh.spinOnce();
 delay(100);
```

Por último, se llama a ros::spinOnce() cuando se han finalizado todas las comunicaciones. Se ha introducido un "delay" o retardo de 0.1 segundos para que el programa no se bloquee y se reciban los datos correctamente.

Para poder visualizar los resultados son necesarios ciertos comandos que se muestran en el apartado de "Resultados".

# 5. Resultados

En este apartado se mostrarán los resultados de la implementación que previamente se ha explicado. Comentar que no ha sido posible integrar el sensor en el vehículo y obtener datos reales de la vía, por lo que el experimento se ha realizado en un coche pasando por encima de tramos de carretera en los que hay grietas.

Por lo tanto, los resultados que se aportan en la presente memoria no son datos reales, pero sí son comparables, pues en la vía cuando el vehículo pasase por encima de algún defecto se producirá un pico de vibración al estar colocado el sensor en el soporte de las ruedas como se mostrará en los próximos apartados.

En definitiva, en este apartado se mostrarán los resultados analíticos y gráficos de los casos anteriores cuando el sensor se encuentra con algún defecto y cómo interpretar dichos resultados.

## 5.1. Valores iniciales

Para obtener los resultados de este apartado, primeramente, se sube el código a la placa Arduino (Fig. 15).


*Fig. 15. Subir código a la placa Arduino*

La forma de ver los datos obtenidos es a través del Monitor Serie de Arduino (Fig. 16).

*Fig. 16. Botón para acceder al Monitor Serie*

Es necesario configurar la velocidad establecida en el código, en este caso 38400 baudios (Fig. 17).

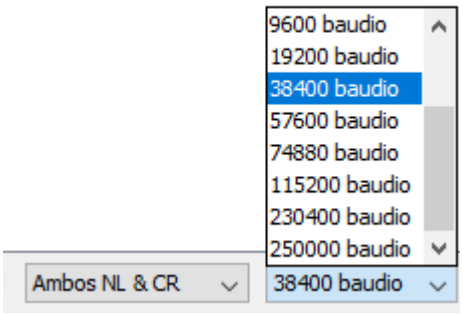


*Fig. 17. Establecimiento velocidad puerto serie*

El paso anterior es muy importante, pues si se establece una velocidad diferente a la indicada en el código no se obtendrá ningún dato, como se indica en la Fig. 18.

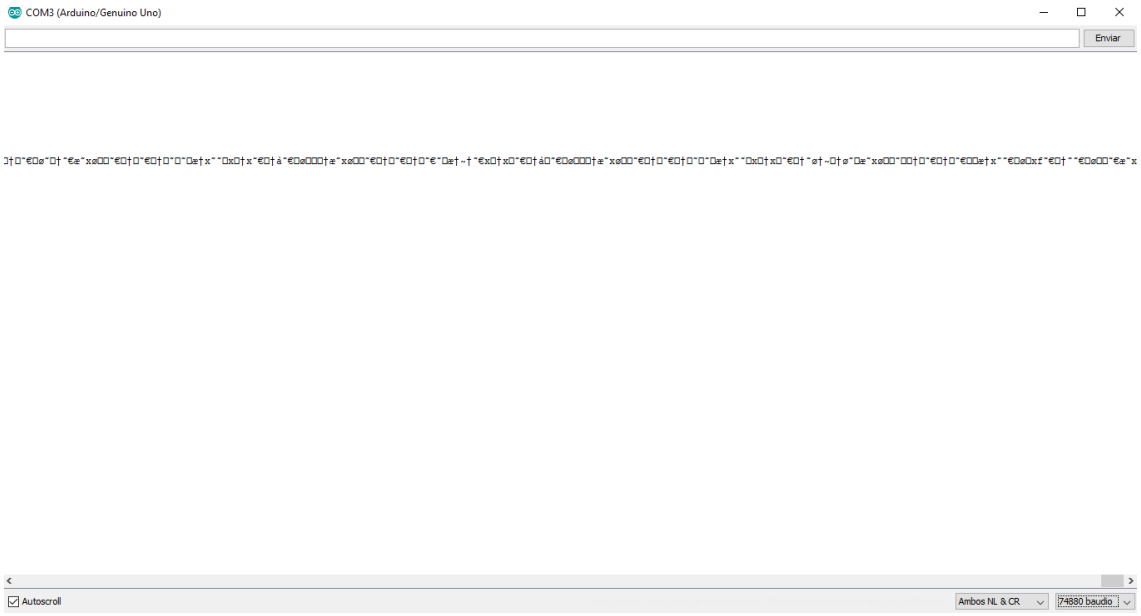


*Fig. 18. Error en la lectura de datos*

Una vez realizado los pasos anteriores, se empiezan a recibir valores por el Monitor Serie de forma continua. Estos datos han sido exportados a la tarjeta microSD y han sido analizados y representados con Excel. Los datos obtenidos de un defecto en la carretera son los representados en la Tabla 1.

| ax | ay | az | gx | gy | gz |
|---|---|---|---|---|---|
| -3072 | -3100 | 15236 | -967 | -297 | 1977 |
| -2028 | -3060 | 15576 | -871 | -560 | 2755 |
| -2556 | -3024 | 14872 | -520 | -1484 | 3555 |
| -2820 | -2936 | 16072 | -854 | -2277 | 4459 |
| -2780 | -2588 | 15892 | -1172 | -2385 | 5020 |
| -1948 | -1576 | 15456 | -953 | -2829 | 5060 |
| -1860 | -1384 | 16820 | -1091 | -3449 | 5087 |
| -2400 | -2268 | 16304 | -620 | -3220 | 5314 |
| -2788 | -1484 | 17772 | -410 | -2112 | 5775 |
| -2568 | -1688 | 17304 | 347 | 340 | 6258 |
| -2700 | -1948 | 16640 | 907 | 1095 | 6580 |
| -1396 | -1552 | 17268 | 1874 | 3304 | 6626 |
| -2236 | -2720 | 21156 | 5476 | 6143 | 3849 |
| -13980 | 4336 | 16384 | 4714 | 8597 | 134 |
| -10588 | -4556 | 11908 | 745 | 5411 | 1920 |
| -4472 | 152 | 8560 | -1902 | 2133 | 2266 |
| -3228 | -1520 | 16060 | -3510 | -6503 | 130 |
| -2808 | -2128 | 22444 | -4060 | -4470 | 700 |

*Tabla 1. Lectura de valores iniciales*

Como puede observarse, no se puede sacar nada en claro de estos datos. No están en sistema internacional, por lo que no son entendibles, aunque se puede apreciar que se produce un pico. Si se representan gráficamente se obtiene lo indicado en la Fig.19 (aceleración) y en la Fig. 20 (velocidad angular), ambas en función del tiempo, medido en milisegundos.

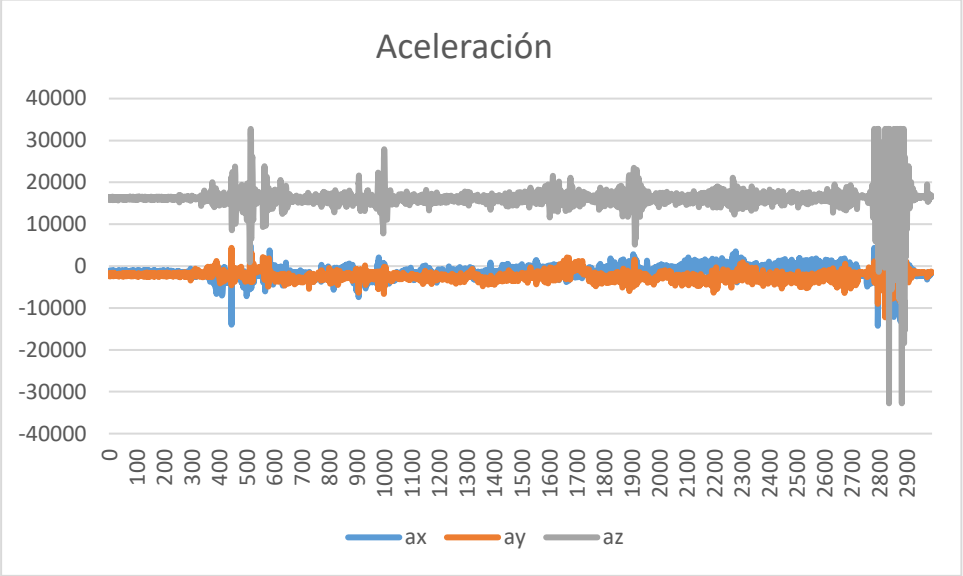


*Fig. 19. Aceleraciones de valores iniciales*

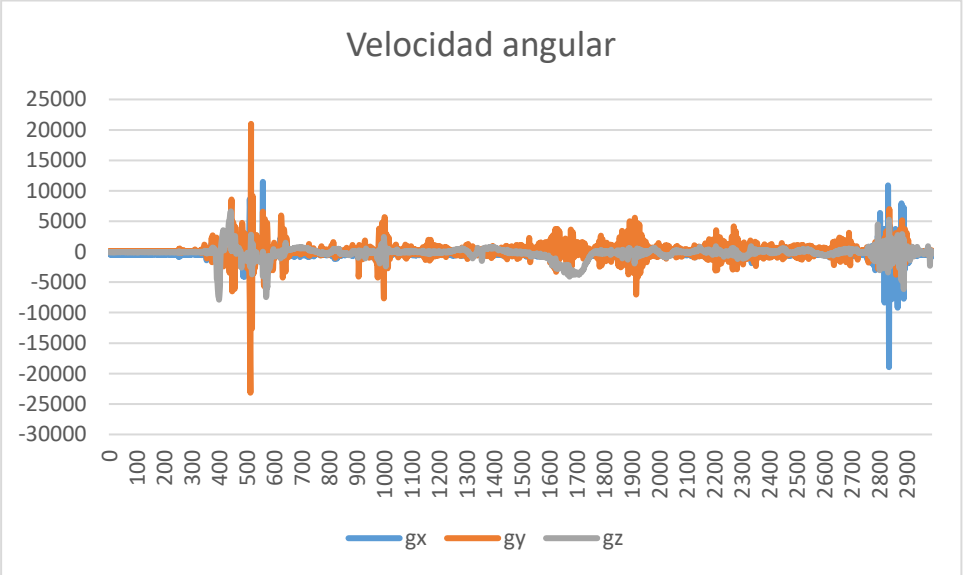


*Fig. 20. Velocidades angulares de valores iniciales*

En esta prueba se han encontrado dos grietas: una al principio, donde se aprecian dos picos (ruedas delanteras y traseras) y al final otros dos picos, producidos también por las ruedas.

## 5.2. Valores escalados

Para este apartado se vuelven a realizar los pasos realizados en el apartado anterior, y se obtienen los resultados de los que se muestra una parte en la Tabla 2:

| ax | ay | az | gx | gy | gz |
|---|---|---|---|---|---|
| -1 | -1,21 | 9,89 | -2 | 2 | -1 |
| -0,04 | -5,18 | 10,07 | -28 | 46 | 51 |
| -0,6 | -0,4 | 4,25 | 9 | 1 | 12 |
| -0,24 | -0,4 | 0,65 | -21 | 5 | 6 |
| 0,77 | -1,48 | 6,79 | -6 | 6 | -6 |
| -0,37 | -1,31 | 6,48 | -10 | -12 | -7 |
| 1,25 | -2,97 | 11,42 | 2 | -8 | -9 |
| 0,69 | 0,8 | -2,55 | -35 | -12 | -3 |
| -3,48 | -0,01 | 3,19 | -46 | 6 | 10 |
| -0,42 | -5,4 | 19,62 | -57 | 18 | 27 |
| -0,95 | -0,14 | 12,7 | -43 | 13 | 19 |
| -1,62 | 0,81 | 3,08 | -38 | -19 | 13 |
| 0,28 | -1,2 | 15,74 | 30 | 3 | -9 |
| -4,45 | 2,48 | 15,93 | 61 | -11 | 15 |
| -0,1 | -2,79 | 1,54 | 2 | 6 | 18 |
| 0,1 | 0,74 | 11,54 | 15 | -14 | 8 |
| -3,26 | 0,53 | 7,91 | -38 | -35 | -20 |
| -9,59 | 3,12 | 14,13 | 27 | 31 | 39 |
| -3,74 | 4,11 | 19,62 | 52 | 1 | 42 |
| -1,27 | -2,35 | 19,47 | 49 | 7 | 4 |
| 1,45 | -1,99 | 4,68 | 26 | -65 | -13 |
| 1,43 | 3,35 | 8,86 | 16 | -17 | 11 |
| 3,69 | 0,23 | 10,66 | -18 | -13 | -12 |

*Tabla 2. Lectura de valores escalados*

El valor de la aceleración en el eje z tiene que ser cercana a 9,8 cuando el sensor está en reposo (valor de la aceleración de la gravedad). Cuando se produce un cambio brusco en este valor y en el resto es cuando se detecta la grieta. Si se representan gráficamente, obtenemos en la Fig. 21 la aceleración (en $m/s^2$) y en la Fig. 22 la velocidad angular (en $^o/s$) frente al tiempo, medido en milisegundos.

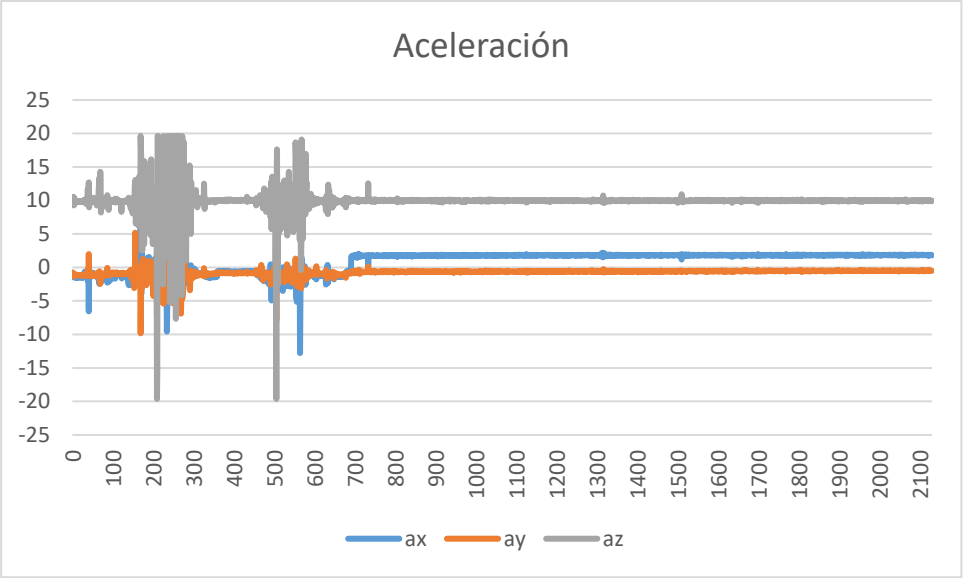


*Fig. 21. Aceleraciones de valores escalados*

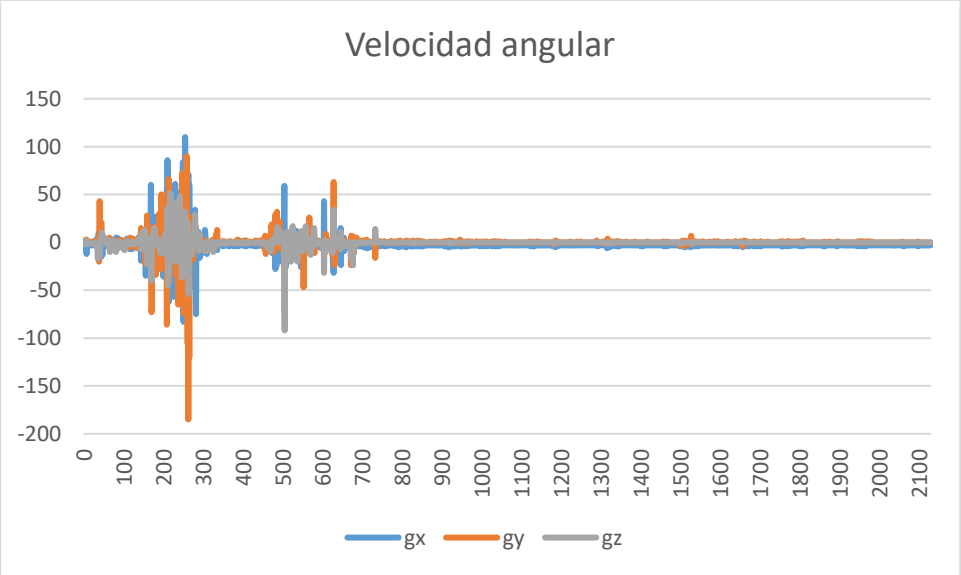


*Fig. 22. Velocidades angulares de valores escalados*

De igual manera, los picos de vibración representan la detección de una grieta. Hay dos, primero con la vibración que se produce cuando las ruedas delanteras atraviesan la grieta y el segundo cuando lo hacen las ruedas traseras.

## 5.3. Lectura de valores en ROS

Para poder visualizar los datos, se deben realizar los siguientes pasos. Primero, se debe inicializar el *roscore* [35], que es el conjunto de recursos necesario para poder usar ROS. Inicializa el master, necesario para la comunicación entre los nodos, como se ha visto anteriormente. Para inicializar *roscore* se abre una ventana de comandos y se introduce “roscore”, como se indica en la Fig. 23.

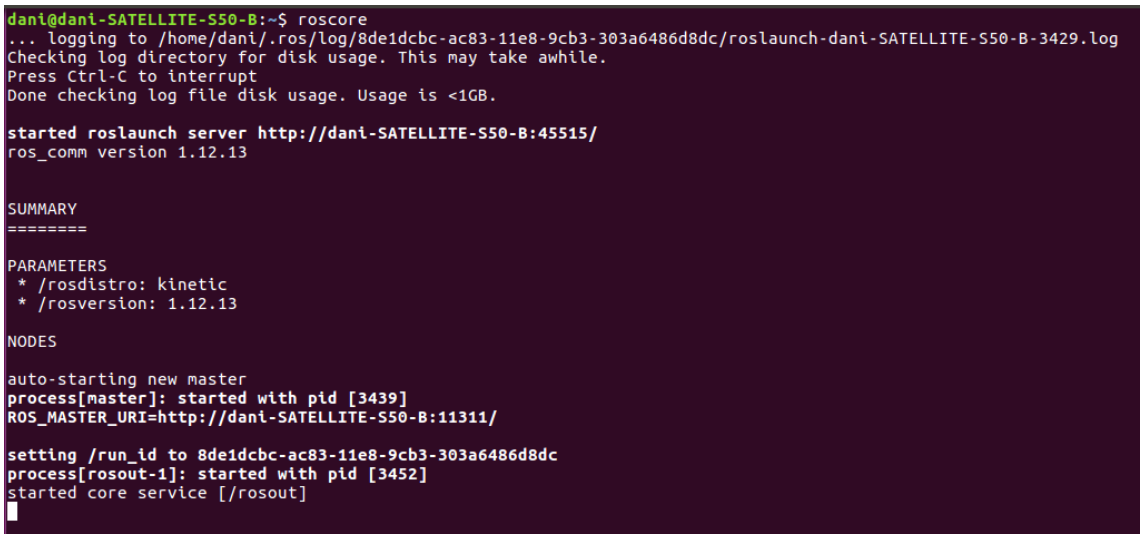


*Fig. 23. Lanzamiento de "roscore"*

En otra ventana de comando se debe introducir el código indicado en la Fig. 24. para inicializar la aplicación "rosserial client" y permitir así que Arduino publique mensajes en ROS.

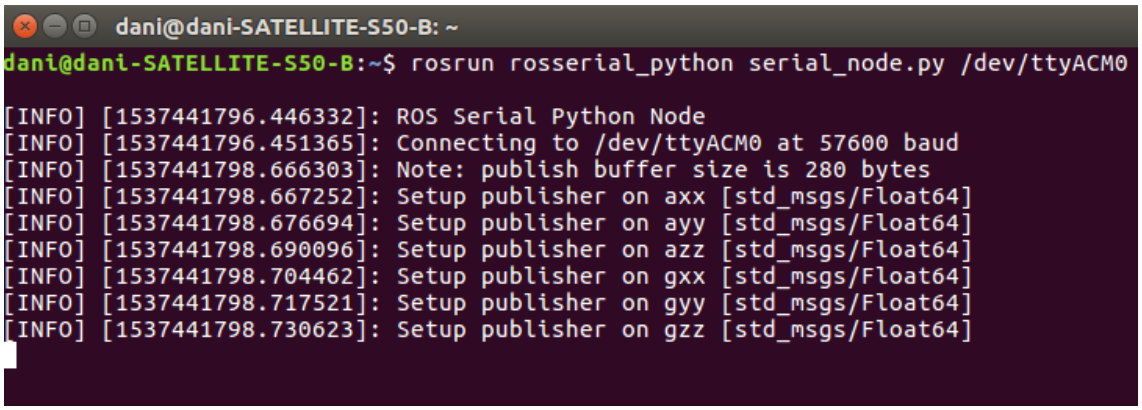


*Fig. 24. Inicialización de "rosserial" con "rosrun"*

En la última parte de la primera línea se indica el puerto USB al que está conectado la placa Arduino. Si no se especifica, se inicializa por defecto a una velocidad de 57600 baudios.

Por último, en otra ventana de comandos se debe introducir "rostopic echo" seguido del topic que queremos visualizar de todos los creados. Los topics se muestran al inicializar "rosserial" (Fig. 24). Como ejemplo, en la Fig. 25 se ha inicializado el topic azz, donde se publican los valores del sensor de la aceleración en el eje z cuando el vehículo se encuentra en reposo (aceleración en el eje z cercana a 9,8 $m/s^2$).

```
dani@dani-SATELLITE-S50-B:~$ rostopic echo azz
data: 10.0423173904
---
data: 9.94651699066
---
data: 9.88903617859
---
data: 10.1261434555
---
data: 9.95130634308
---
data: 9.8626909256
---
data: 10.0375270844
---
data: 10.0087871552
---
data: 9.92496109009
---
data: 9.88664150238
---
data: 10.0111818314
---
data: 9.98723220825
---
data: 9.95370197296
---
data: 9.99920654297
---
data: 9.93214607239
---
data: 9.94891166687
---
data: 9.96807193756
---
data: 10.0542926788
---
data: 10.0039968491
---
data: 9.97286128998
---
data: 9.86029624939
---
```

*Fig. 25. Inicialización del topic "azz"*

Los datos que se obtienen cuando se encuentra un defecto son los mismos que en el apartado anterior, simplemente son publicados en ROS. En este apartado solo se muestran los resultados gráficos de las aceleraciones y velocidades angulares con el comando *rqt_plot* [36].

En una ventana de comandos nueva se introduce dicho comando y se abre una ventana nueva en la que se selecciona los topics que se desea graficar, como se puede ver en la Fig. 26.

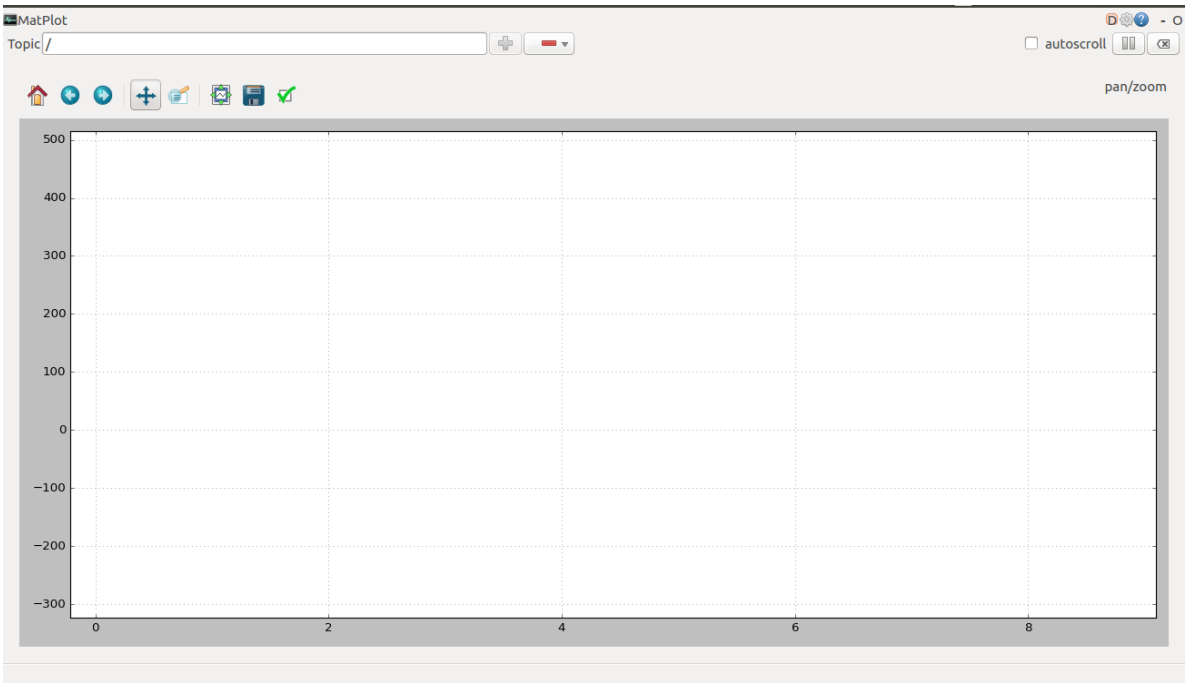


*Fig. 26. rqt_plot*

En la Fig. 27 se muestran las aceleraciones cuando se detecta una grieta.

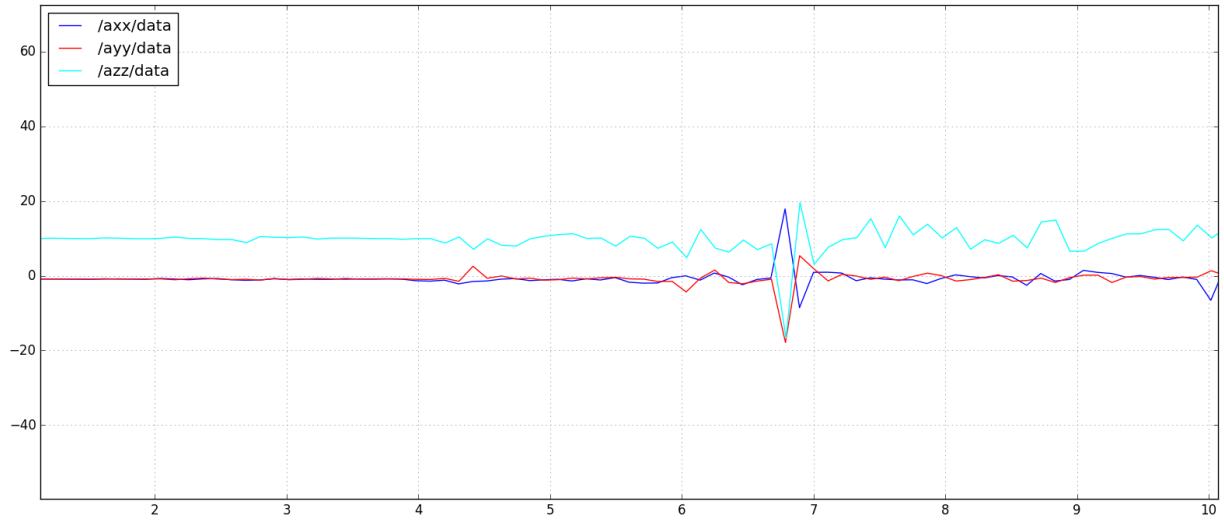


*Fig. 27. Aceleraciones visualizadas con ROS*

Así mismo, cuando se produce un pico de vibraciones las velocidades angulares quedan como lo representado en la Fig. 28.

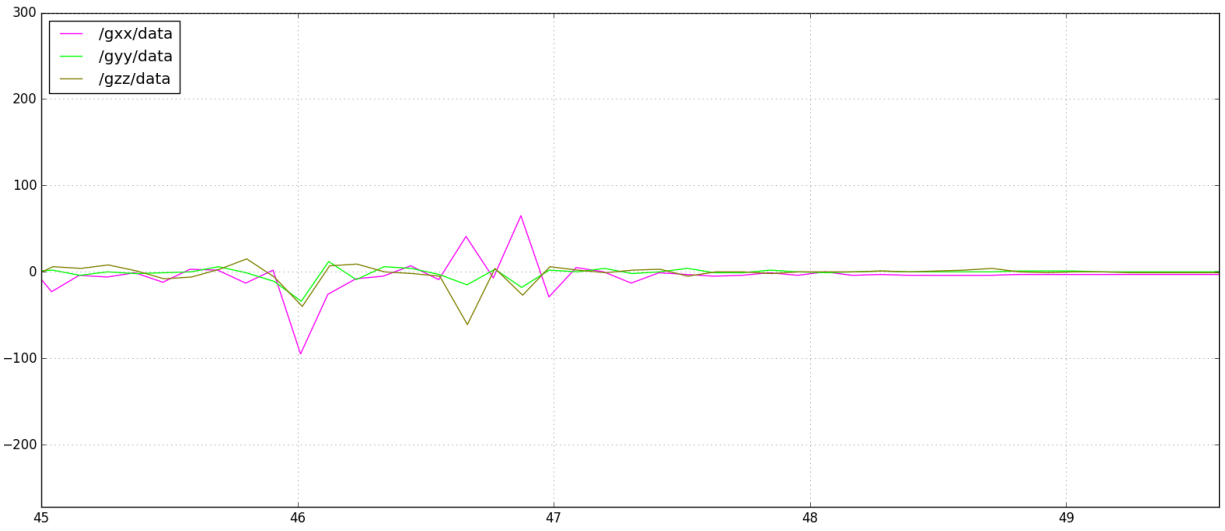


*Fig. 28. Velocidades angulares visualizadas con ROS*

En conclusión, una vez publicado los datos en ROS, el software de FerroDron combinará todos los valores para proporcionar datos sobre el estado de la vía y visualizar en que punto kilométrico de la vía se produce, con el fin de georreferenciar el defecto.

# 6. Entorno socioeconómico

## 6.1. Impacto socioeconómico

En este apartado se argumenta el impacto social, económico y medioambiental de la aplicación del resultado del proyecto.

- **Impacto social:** con la implantación del proyecto se prevé una mejora de los servicios de inspección y cuidado de la infraestructura ferroviaria, dado que se puede obtener con exactitud el estado de un raíl, lo que disminuye el tiempo de respuesta a corregir fallas en las vías y se traduce en una mejora de la calidad del servicio y mayor seguridad.
- **Impacto económico:** lo que se busca con este proyecto es principalmente la autonomía y la automatización en la tarea de inspección de las vías. Esto significa que se crearán algunos puestos de trabajo, como los de control del sistema, o ingenieros y técnicos de mantenimiento. Pero por otra parte el número de trabajadores se ve reducido pues no es necesario que cierto número de trabajadores traslade la maquinaria por el eje ferroviario en busca de defectos.
- **Impacto medioambiental:** dado que el sensor se instalaría en FerroDron que circula por las vías ya existentes en horario nocturno, no es necesaria la creación de nueva infraestructura ferroviaria en el proceso. Además, se trata de un vehículo eléctrico por lo que no es contaminante, por lo que el impacto medioambiental es nulo.

## 6.2. Duración de las tareas

En este capítulo se presenta la duración de las tareas que se han llevado a cabo en este proyecto. Como se muestra en la Tabla 3, la duración del mismo ha sido aproximadamente de 4 meses.

| | Tarea | Inicio | Fin |
|---|---|---|---|
| **1** | Estudio y elección de tecnologías disponibles | 01/06/2018 | 10/06/2018 |
| **2** | Aprendizaje programación en Arduino | 18/06/2018 | 25/06/2018 |
| **3** | Aprendizaje programación ROS | 26/06/2018 | 12/07/2018 |
| **4** | Aprendizaje programación rosserial_arduino | 16/07/2018 | 23/07/2018 |
| **5** | Diseño del circuito | 23/07/2018 | 24/07/2018 |
| **6** | Implementación código valores iniciales | 25/07/2018 | 25/07/2018 |
| **7** | Implementación código valores S.I. | 26/07/2018 | 27/07/2018 |
| **8** | Integración con ROS | 30/07/2018 | 03/08/2018 |
| **9** | Implementación código con rosserial_arduino | 13/08/2018 | 17/08/2018 |
| **10** | Pruebas experimentales | 20/08/2018 | 25/08/2018 |
| **11** | Corrección de errores | 26/08/2018 | 26/08/2018 |
| **12** | Pruebas finales | 27/08/2018 | 28/08/2018 |
| **13** | Redacción de la memoria | 25/07/2018 | 16/09/2018 |

*Tabla 3. Duración de las tareas llevadas a cabo en el proyecto*

A continuación se procede a realizar una breve descripción de las tareas anteriores para un mejor entendimiento:

- **Tarea 1:** búsqueda de los programas, sensores, y opciones posibles para la realización del proyecto
- **Tarea 2:** lectura de documentación y búsqueda de información y código para aprender como programar en Arduino.
- **Tarea 3:** lectura de documentación y búsqueda de información y código para entender el funcionamiento de ROS y como programar en el mismo.
- **Tarea 4:** una vez entendido ROS se procedió al entendimiento de la integración de ROS en Arduino con el paquete rosserial_arduino.
- **Tarea 5:** diseño y montaje del circuito del proyecto con el sensor y el lector de tarjetas microSD, realizando las soldaduras necesarias.
- **Tarea 6:** implementación del código de valores iniciales obtenido de DFRobot.
- **Tarea 7:** implementación del código para la obtención de valores en sistema internacional.
- **Tarea 8:** pruebas para determinar la opción definitiva entre la integración del sensor directamente con ROS o con el paquete rosserial_arduino.
- **Tarea 9:** implementación del código para publicar los valores obtenidos por el sensor en ROS con el paquete rosserial_arduino.

- **Tarea 10:** realización de pruebas de detección de defectos en el coche.
- **Tarea 11:** tras las primeras pruebas se corrigieron ciertos errores en el código que impedían alcanzar el objetivo del proyecto.
- **Tarea 12:** tras la corrección de errores se realizaron las pruebas finales que recogen los datos que se muestran en la memoria.
- **Tarea 13:** realización de la presente memoria.

En la Fig. 29 se adjunta el Diagrama de Gantt de la duración el proyecto.

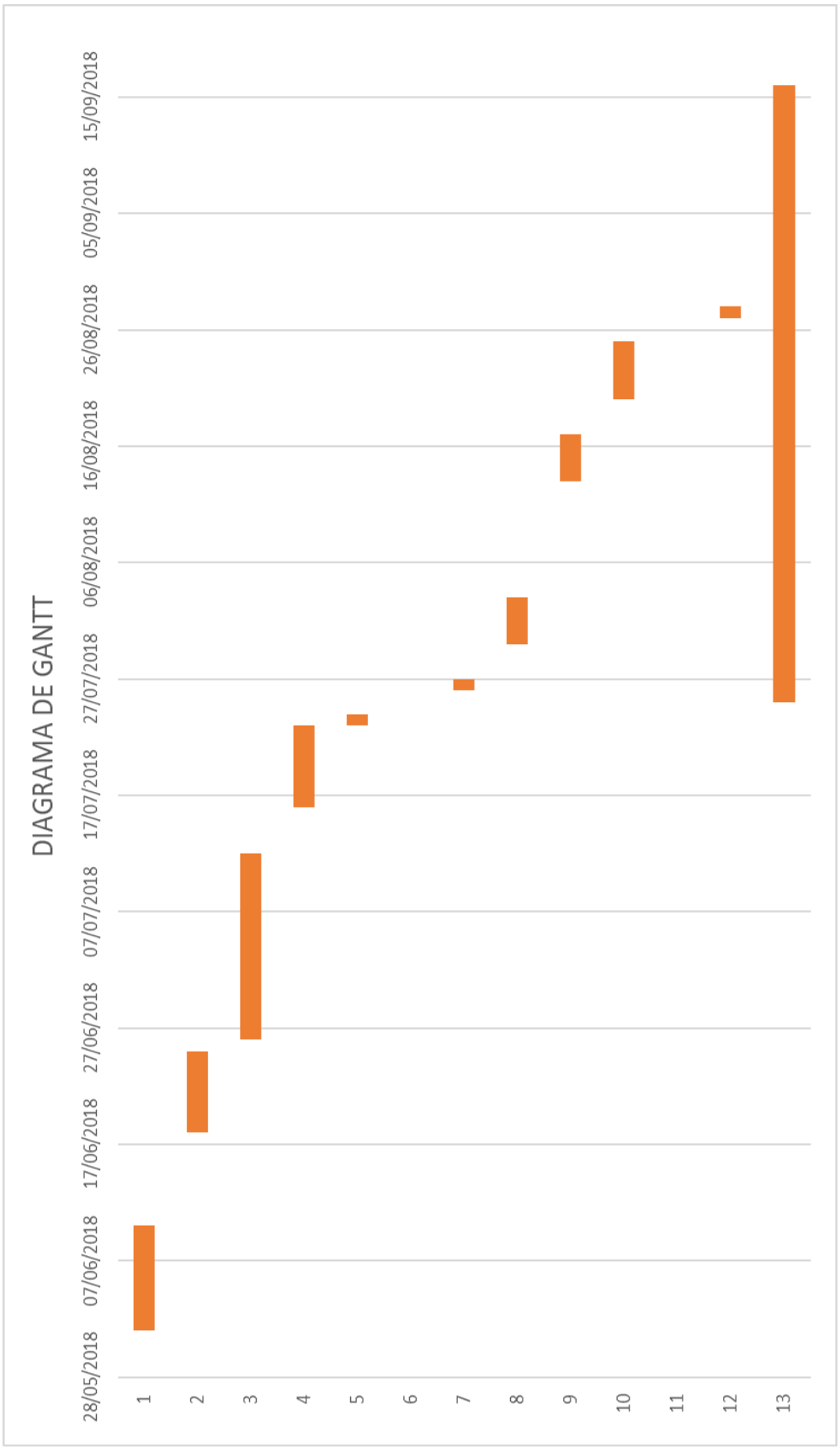


*Fig. 29. Diagrama de Gantt del proyecto*

## 6.3. Presupuesto

Para la elaboración del presupuesto de la realización del proyecto se han tenido en cuenta tanto la mano de obra como los medios tecnológicos usados para su creación, y son los que se exponen a continuación:

- **Hardware:** los dispositivos electrónicos que se han usado han sido un ordenador portátil, una placa Arduino y un sensor inercial MPU6050. Además para la realización de la memoria se ha usado un módulo lector de tarjetas microSD y una placa protoboard. Los cables usados para las conexiones no se tienen en cuenta por ser de valor despreciable. Se considera que la vida útil de un ordenador son 7 años para hacer cálculos de costes, y la vida útil de la electrónica 4 años. El precio medio de un ordenador se considera 800 €, la placa Arduino 20 €, el sensor MPU6050 8,42 €, la placa protoboard 7 €, y el módulo lector 3,50 €. En definitiva, el gasto total en hardware ha sido **43,72 €**.

  $$Hardware = 850\ € \ x\ \frac{4\ meses}{7\ años\ x\ 12\ meses/año} + (20\ € + 8{,}42\ € + 7€ + 3{,}50\ €) x \frac{4\ meses}{4\ años\ x\ 12\ meses/año} = 43{,}72\ €$$

- **Software:** todo el software utilizado era gratuito excepto Microsoft Excel y Microsoft Word, utilizados para analizar y graficar los datos y la redacción de la presente memoria. A pesar de ello, se disponía de una licencia educativa por un convenio existente entre la Universidad Carlos III de Madrid y Microsoft. Por lo tanto, los costes de software son nulos.
- **Salarios:** para estos costes se han tenido en cuenta las tablas salariales proporcionadas por la Universidad Carlos III de Madrid, y se calcula el presupuesto de mano de obra que se adjunta en la Tabla 4.

| Trabajador | Categoría | Coste por hora | Horas dedicadas | Total |
|---|---|---|---|---|
| **Daniel Segovia Magaz** | Recién titulado | 23,43 € | 350 | 8.200,5 € |
| **Aurelio Ponz Vila** | Titulado doctor | 35,33 € | 20 | 706.6 € |

*Tabla 4. Salarios*

Por lo que el presupuesto de mano de obra total es **8907.1 €**

Por lo tanto, el presupuesto total de implantación del proyecto es de **OCHO MIL NOVECIENTOS CINCUENTA CON OCHENTA Y DOS EUROS (8.950,82 €).**

# 7. Conclusiones y trabajo futuro

La inspección y reparación de las vías ferroviarias es una tarea que implica un traslado de maquinaria, además del traslado humano que maneja dicha maquinaria. Además es un proceso que necesita de una alta cantidad de tiempo. El presente proyecto busca que esta tarea sea más liviana y eficaz, mediante la detección de grietas en las vías con el propio paso del vehículo ferroviario.

Sin embargo, aunque el proyecto ha sido desarrollado, no ha podido ser implantado en FerroDron, por lo que en un futuro trabajo sería la primera tarea a solventar. Los datos recogidos han sido probando el sensor en un coche, que absorbe las vibraciones y no se pueden detectar datos exactos. Lo que sí es cierto es que el pico de vibración se produciría de igual manera cuando el vehículo ferroviario se encontrase con una grieta en la vía.

Otra posible mejora del trabajo podría ser la aplicación de uno de los filtros mencionados en el proyecto para la eliminación de ruido, una vez que se probase en el vehículo ferroviario, pues este produciría más vibraciones y sí podría inducir a errores en la detección de imperfecciones. En las gráficas representadas se ha podido observar que cuando el vehículo se encuentra en reposo absoluto se producen ciertos picos de vibración debido al ruido mencionado.

Por otra parte, una alternativa al sensor utilizado podría ser Pixhawk [37]. Se trata de un proyecto de hardware abierto principalmente usado en la actualidad para el control de drones. Pixhawk posee dos IMUs, que serían las usadas para la realización del proyecto. En lugar de usar dos sensores como los usados en el trabajo, se utilizaría una Pixhawk que estaría instalada en la plataforma del vehículo ferroviario, pero el chasis absorbe las vibraciones y podría no suministrar datos reales. Sin embargo, con la Pixhawk pueden obtenerse datos de todo el vehículo y no de cada uno de los raíles independientemente como se ha hecho en el proyecto.

En conclusión, esta primera versión del proyecto tiene ciertas mejoras, pero se han cumplido los objetivos propuestos mediante la recogida de datos del sensor y la publicación de los mismos en los topics de ROS creados y se ha hecho un análisis de cuándo existiría una grieta según las gráficas de vibraciones.

# Bibliografía

[1] Artículo 25, Capítulo I, Título VII de la Orden FOM/167/2015, de 6 de febrero, por la que se regulan las condiciones para la entrada en servicio de subsistemas de carácter estructural, líneas y vehículos ferroviarios. *Boletín Oficial del Estado, núm. 35*, de 10 de febrero de 2015, páginas 10757 a 10788. [En línea]. Disponible en: https://www.boe.es/boe/dias/2015/02/10/pdfs/BOE-A-2015-1234.pdf [Accedido: 25-07-2018].

[2] "¿Qué es un acelerómetro?", *Omega.* [En línea]. Disponible en: https://es.omega.com/prodinfo/acelerometro.html [Accedido: 26-07-2018].

[3] A. Sabán. "Cómo funcionan el acelerómetro y el giroscopio de los móviles", *Hipertextual*, 26-08-2016. [En línea]. Disponible en: https://hipertextual.com/2016/08/acelerometro-giroscopio [Accedido: 27-07-2018].

[4] "¿Qué es una IMU?", *Robdos Team Underwater Robotics.* [En línea]. Disponible en: http://www.robdosteam.com/wp-content/uploads/2016/12/08.IMU_.pdf [Accedido: 27-07-2018].

[5] Apuntes de la asignatura: "Tecnología de materiales" del Grado en Tecnologías Industriales. *Universidad Carlos III de Madrid*. Curso 2017/2018.

[6] C. Rimoldi y L.M. Mundo. "Ensayo no destructivo por método de ultrasonido". Apunte de clase de Cátedra: Ensayos no destructivos. *Universidad Nacional de la Plata*. Año 2012. [En línea]. Disponible en: http://www.aero.ing.unlp.edu.ar/catedras/archivos/Apunte%20Ultrasonido%202012.pdf [Accedido: 30-07-2018].

[7] "Process Compensated Resonant Testing", *Vibrant Corporation & Magnaflux Quasar*. [En línea]. Disponible en: http://www.vibrantndt.com/wp-content/uploads/Magnaflux-Vibrant-Quality-NDT-paper1.pdf [Accedido: 02-08-2018].

[8] S. Fernández Matey. "*Clasificación y análisis de la evolución de la respuesta modal de un eje ferroviario con grietas*". Proyecto fin de carrera. Escuela Politécnica Superior de la Universidad Carlos III de Madrid. Leganés, España, 2013. [En línea]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/18486/PFC_Sergio_Fernandez_Matey.pdf?sequence=1&isAllowed=y [Accedido: 03-08-2018].

[9] M. González Martínez. *"Registro y análisis vibratorio de un eje ferroviario a escala para defectología".* Proyecto fin de carrera. Escuela Politécnica Superior de la Universidad Carlos III de Madrid. Leganés, España, 2012. [En línea]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/22658/PFC_miguel_gonzalez_martinez_2014.pdf?sequence=1&isAllowed=y [Accedido: 03-08-2018].

[10] "FerroDron", *Universidad Carlos III de Madrid*. [En línea]. Disponible en: http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/IntelligentSystemsLab/research/public/ferrodron [Accedido: 15-08-2018].

[11] Expediente RTC-2015-3953-4, Disposición 1991, Sección III, de la Resolución de 17 de febrero de 2016, de la Secretaría de Estado de Investigación, Desarrollo e Innovación, por la que se publican las ayudas concedidas en la convocatoria Retos-Colaboración 2015, convocadas por Resolución de 30 de diciembre de 2014. *Boletín Oficial del Estado, núm. 49*, de 26 de febrero de 2016, páginas 15950 a 16027. [En línea]. Disponible en: https://www.boe.es/boe/dias/2016/02/26/pdfs/BOE-A-2016-1991.pdf [Accedido: 15-08-2018].

[12] Información cedida por la Universidad Carlos III de Madrid sobre el proyecto FerroDron.

[13] "6-DOF Sensor-MPU6050", *DFRobot*. [En línea]. Disponible en: https://www.dfrobot.com/wiki/index.php/6_DOF_Sensor-MPU6050_(SKU:SEN0142) [Accedido: 02-06-2018].

[14] "¿What is Arduino?", *Arduino*. [En línea]. Disponible en: https://www.arduino.cc/en/Guide/Introduction [Accedido: 06-08-2018].

[15] "Arduino UNO Rev3", *Arduino*. [En línea]. Disponible en: https://store.arduino.cc/arduino-uno-rev3 [Accedido: 06-06-2018].

[16] "BASIC Stamp 1 Microcontroller Module", *Parallax Inc.* [En línea]. Disponible en: https://www.parallax.com/product/bs1-ic [Accedido: 06-06-2018].

[17] "BasicX-24 Description", *Netmedia.* [En línea]. Disponible en: http://www.basicx.com/Products/BX-24/bx24overview.htm [Accedido 25-08-2018].

[18] V. García. "Introducción al I2C-Bus", *Electrónica Práctica Aplicada*, (2012). [En línea]. Disponible en: https://www.diarioelectronicohoy.com/blog/introduccion-al-i2c-bus [Accedido: 20-08-2018].

[19] "Wire Library", *Arduino.* [En línea]. Disponible en: https://www.arduino.cc/en/Reference/Wire [Accedido: 20-08-2018].

[20] "Tutorial de Arduino y MPU-6050", *Roboblogs*, 15-10-2014. [En línea]. Disponible en: https://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/ [Accedido: 12-06-2018].

[21] "Diseño y utilización de filtros de Kalman en MATLAB y Simulink", *Mathworks*. [En línea]. Disponible en: https://es.mathworks.com/discovery/filtros-kalman.html [Accedido: 29-08-2018].

[22] L. Llamas, "Medir la inclinación con IMU, Arduino y filtro complementario", *El blog de Luis Llamas*, 07-09-2016. [En línea]. Disponible en: https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/ [Accedido: 15-06-2018].

[23] "Lector Micro-SD Catalex", *Patagoniatecs*. [En línea]. Disponible en: https://saber.patagoniatec.com/2016/01/lector-microsd-catalex/ [Accedido: 25-08-2018].

[24] "About ROS", *ROS*. [En línea]. Disponible en: http://www.ros.org/about-ros/ [Accedido: 28-06-2018].

[25] "Master", *ROS.* [En línea]. Disponible en: http://wiki.ros.org/Master [Accedido: 02-07-2018].

[26] "Nodes", *ROS.* [En línea]. Disponible en: http://wiki.ros.org/Nodes [Accedido: 02-07-2018].

[27] "Messages", *ROS.* [En línea]. Disponible en: http://wiki.ros.org/Messages [Accedido: 02-07-2018].

[28] "Topics", *ROS.* [En línea]. Disponible en: http://wiki.ros.org/Topics [Accedido: 02-07-2018].

[29] "rosserial_arduino", *ROS.* [En línea]. Disponible en: http://wiki.ros.org/rosserial_arduino [Accedido: 15-07-2018].

[30] "SD Library", *Arduino*. [En línea]. Disponible en: https://www.arduino.cc/en/Reference/SD [Accedido: 21-07-08].

[31] "SPI Library", *Arduino.* [En línea]. Disponible en: https://www.arduino.cc/en/Reference/SPI [Accedido: 22-07-08].

[32] L. Llamas. "Leer y escribir en una tarjeta SD o microSD con Arduino", *El blog de Luis Llamas*, 16-10-2016. [En línea]. Disponible en: https://www.luisllamas.es/tarjeta-micro-sd-arduino/ [Accedido: 25-07-2018].

[33] "MPU-6000 and MPU-6050 Product Specification", *Invensense Inc.* [En línea]. Disponible en: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf [Accedido: 16-08-2018].

[34] "Hello World (example Publisher)", *ROS*. [En línea]. Disponible en: http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World [Accedido: 15-07-2018].

[35] "roscore", *ROS*. [En línea]. Disponible en: http://wiki.ros.org/roscore [Accedido: 20-07-2018].

[36] "rqt_plot", *ROS*. [En línea]. Disponible en: http://wiki.ros.org/rqt_plot [Accedido: 20-07-2018]

[37] "Pixhawk 4", *Pixhawk*. [En línea]. Disponible en: https://docs.px4.io/en/flight_controller/pixhawk4.html [Accedido: 01/09/2018].

# Summary

## Introduction

Detect non-superficial fissures or imperfections in any material is not an easy task. Is needed some methods like ultrasonic testing or X-Ray to carry out the aforementioned purpose. In the railway field, to find out the condition of the train rail before a failure is produced is complicated. Furthermore, it has an elevated wage cost, because the railway maintenance team must transport the required equipment though the railroad track.

The purpose of this work is to facilitate this task. Using the existing project FerroDron, which is based on an automatic railway drone, an inertial sensor could be installed to take data from the railroad track, working together with Arduino and ROS with the objective of detecting and visualizing the train's vibrations and being able to find out where the defects could be.

The main objective of this project is to install an inertial sensor in the wheel support of the railway vehicle with the aim of detecting in an easy way the condition of the railroad track, thanks to the vibrations produced by the vehicle itself. For that, an Arduino code has been created, besides several ROS topics where the data collected by the sensor is published.

FerroDron is composed by numerous sensors. ROS will allow us to integrate the inertial sensor with the others, in order to work together with the other sensors of the vehicle (cameras, laser, inclinometers, GPS…). All the sensors publish with ROS their data and the software combine them to collect data from the railroad track.

Summarizing, the objectives of this bachelor thesis are:

- Choice of the inertial sensor.
- Implementation of the Arduino code with the aim of detecting vibrations.
- Integration with ROS.
- Visualization of the collected data and interpretation of these.

There are no regulations about this project, since the vehicle mentioned above circulates at night and it doesn't transport any passengers, so it doesn't have to fulfill the Orden FOM/167/2015 [1], which says that before providing any services of transport, the railway companies must get the security certificate. However, FerroDron must be compatible with the circulations of the railway vehicles.

## State of Art

For the correct understatement of the project, is necessary to explain different sensors that are used to measure vibrations, as well as some technologies that can be used to find defects in materials and other projects in where an accelerometer has been used to detect vibrations in the railroad tracks.

The main sensors used to measure vibrations are accelerometers, gyroscopes and inertial sensors:

- **Accelerometers** are devices able to measure the acceleration of an object [2]. Because of that, is a usual instrument in the measure of vibrations. Nowadays the most common accelerometers used are the three-axes accelerometers, that measure in a microscopic level the accelerations in the X, Y, Z axes.
- **Gyroscopes** are instruments that measure the angular velocity. It can be analogic or electronic, but it is more frequently used the electronic gyroscope. Gyroscopes measure rotations. If the angular velocity is integrated over time the result obtained is the angular displacement, so it is a viable choice to measure vibrations.
- **IMUs (Inertial Measure Units)** [4] are electronic devices that combine one or more accelerometers and gyroscopes, with the aim of measuring velocity and rotation at the same time. Sometimes is also used a magnetometer to collect more accurate measures, generally for locations (because is used as a compass). Definitely, is a much more complete choice in the field of measuring vibrations, because it can obtain more-real measures than one of the sensors working individually, especially in movements in which the accelerometer can't detect movement and the gyroscope can, and vice versa. In the Fig. 30 is shown a 9 DOF (Degrees of Freedom) IMU, which are the three measure axes of the accelerometer, the magnetometer and the gyroscope.


*Fig. 30. Axes of work of a 9 DOF IMU (Figure obtained from [4])*

On the other hand, there is some technologies that we can use to find defects in a material. In this project, we talk about industrial radiography, ultrasonic testing and PCRT.

- **Industrial radiography** can find internal defects in a material when the ionizing radiation goes through it, which generate a visual image of the fault. X-Ray and Gamma-Ray can be used. Industrial radiography is a good choice because it can be used both in ferromagnetic materials and in those that are not. However, is difficult to use in complex geometry parts and it is not a cheap technology.
- **Ultrasonic testing** is a NDT (Non-Destructive Testing) testing. It uses high frequency sound waves to find out and measure the internal defects of a part [5]. Like industrial radiography, ultrasonic testing can be employed in all materials, but it is not able to detect superficial defects and it is not effective in parts with high porosity, because wrong results could be obtained.
- **PCRT (Process Compensated Resonance Testing)** is also a NDT. It measures resonance frequencies. When a defect is found, a change in the resonance frequencies is produced, so a vibration can predict the structural integrity of a part [7]. PCRT can also find the beginning of the fatigue process before the fault has appeared.

Explanations about the design of the current project can't begin without talking about the base project from which arises this bachelor thesis: FerroDron.

**FerroDron** is a project carried out by Carlos III University of Madrid [10]. It proposes the creation of a microlight unmanned vehicle (shown in Fig. 31) whose work is the inspection, maintenance and vigilance of the railway infrastructure [12]. In conclusion, it is an automatic railway drone which monitor the railroad track condition, sending data automatically to a checkpoint in real time.

*Fig. 31. FerroDron (Figure obtained from [12])*

Some functions FerroDron can do, for example, is the construction of a virtual image of the track it goes through thanks to the sensors it has, like artificial vision.

The objective of the project is that the sensor used in the bachelor thesis is placed in this vehicle and that it works together with the rest of the sensor integrated in FerroDron, providing exact data about the condition of the railroad track, since nowadays it captures just the external environment. For that purpose, ROS will be used because the chosen sensor for this project will publish data in there, so in addition with the other sensors they can provide the condition of the railway track.

## Design

The first design choice must be the **sensor choice**. The chosen sensor was an IMU. The main reason is that can make more accurate measures, and since the sensor will be placed in the wheel support if the railway vehicle, the joint work of the accelerometer and the gyroscope will provide more accuracy when detecting imperfections in the railway track. Bearing this in mind, and between hundreds of inertial sensors, the chosen sensor was the **MPU-6050**, by Invensense.

MPU-6050 sensor [13] is a 6 DOF IMU, which combines a 3-axes gyroscope and a 3-axes accelerometer (6 degrees of freedom). Besides, it has an DMP or Digital Motion Processor. The communication with the controller is by I2C bus. Summarizing, the MPU-6050 (Fig. 32) is a complete choice for the project, as long as an appropriate microcontroller is used to stablish I2C communication.

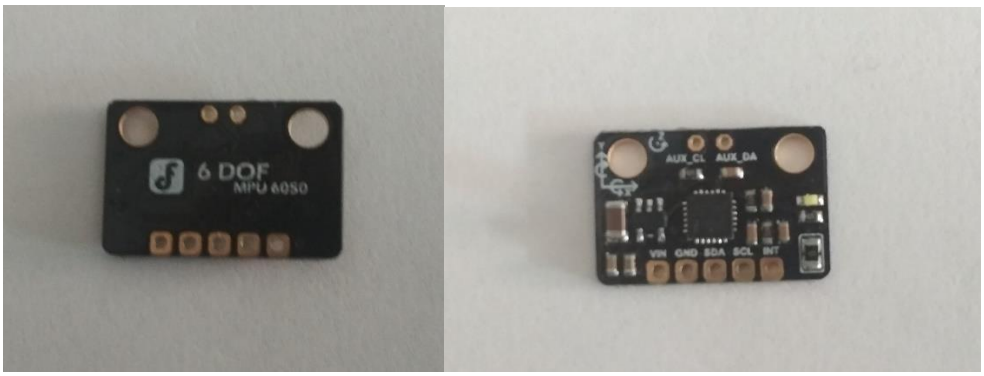


*Fig. 32. MPU-6050 Sensor*

The next choice was the **microcontroller**. The chosen microcontroller was the **Arduino UNO board** [14]. It is a microcontroller board developed by Arduino. Arduino is an open-source platform, so it is simple to find useful code for a project and re-use it. Arduino UNO board is cheap and simple to use, because its programming in Arduino language is very intuitive. In the Fig. 33 is shown the aforementioned board.

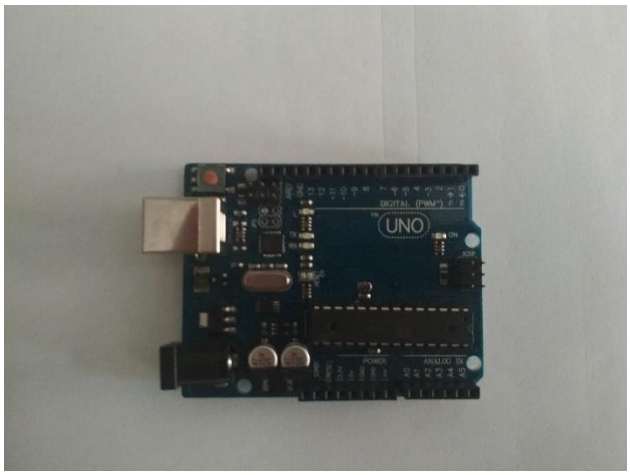


*Fig. 33. Arduino UNO board*

The main reason of the choice is that Arduino UNO is very simple to use. Besides, there are lot of manuals to implement the connection between the MPU-6050 sensor and Arduino. Other reason is that ROS can be integrated in a simple way directly in Arduino.

In the other hand, to collect data from the sensor and analyze it this essay, was necessary to export the data to a Microsoft Excel sheet, in order to visualize this data. For that, a **microSD card lector module for Arduino** was used to export the data to a txt file in the microSD card. This adaptor, developed by Catalex [23] supports cards up to 32 GB, that was the microSD card used. The adaptor is shown in the Fig. 34.

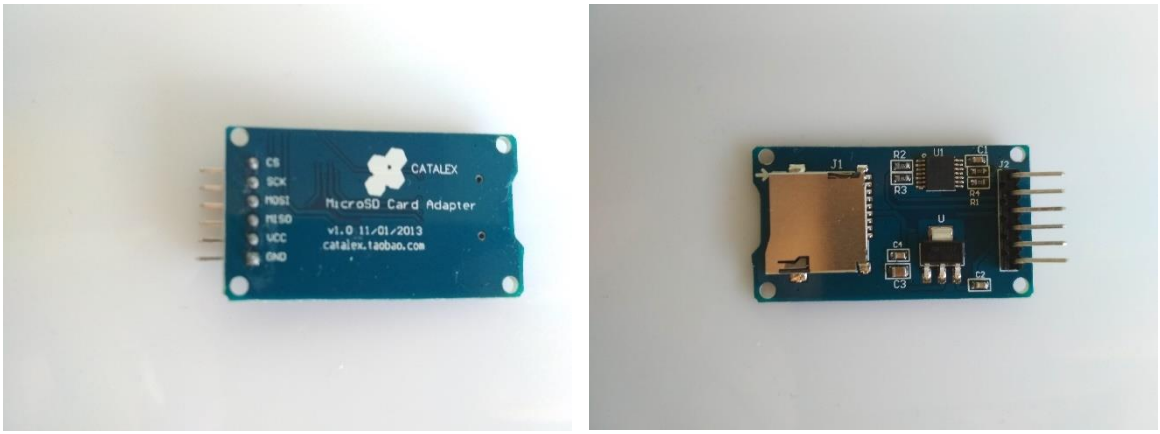


*Fig. 34. MicroSD card lector module for Arduino*

In this project, ROS is used because the sensor must publish the data collected in ROS topics, so the software can combine them with the data obtained from the other sensors and determine the condition of the railroad track. ROS is also an open-source platform so the code found can be re-used. For the integration of the sensor with ROS, **rosserial_arduino** package was chosen [29]. With this, there is the possibility to create nodes and topics directly in the created software in Arduino.

The last design choice was the **OS** (Operating System) that was **Linux**, because the work with ROS is easier in Linux than in Windows.

## Implementation

The devices and programs used for this project are:

**Devices:**

- Laptop computer Toshiba Satellite, late 2014.
  - Intel Core i7 processor, 3.1 GHz
  - 8GB 1600 MHz DDR3L memory.
  - AMD Radeon R7 M260 2GB graphics.
  - 1 TB hard disk.
  - Counts with a partition to change the OS.
  - Main OS: Windows 10
  - Secondary OS (Linux): Ubuntu 16.04

**Programs:**

- Arduino IDE 1.8.6 for Linux: development environment of the necessary code for the IMU and its connection with the Arduino board.
- ROS Kinetic: is not a program but is the platform in where the sensor publishes the collected data.
- Fritzing 0.9.3b.32 for Windows: program which allows to do connection schemes of the Arduino board with a wide variety of electronic components.
- Microsoft Word 2016 for Windows: text editor used for the writing of this essay.
- Microsoft Excel 2016 for Windows: application based on spreadsheets used to analyze the data collected by the sensor and to represent them graphically.

As this a summary, the explanation about the implementation is short. Basically the connections between the MPU sensor, the microSD card lector adaptor and Arduino where implemented, as the code use to measure the acceleration and the angular velocity was created following some indications by DFRobot [13] and Luis Llamas [32]. With this, the results obtained indicated that when a peak of vibration is produced, a superficial defect is detected. Once that was clear, the data was published in 6 different ROS topics (3 for acceleration and 3 for angle velocity) and was represented graphically with the command *rqt_graph*.

## Planification and budget

The total duration of this project was approximately 4 months.

Regarding the budget, it is necessary to bear in mind the software, hardware and wages cost. The software used in this project was an open-source and free software, except Microsoft Office. But the Carlos III University of Madrid has some agreements with educative licenses, so the cost of software was null.

Talking about the hardware, it is needed to bear in mind the cost of a laptot and all the electronic components. In conclusion, the cost of hardware was **43,72 €.**

Finally, about the wage cost, we have to consider each member of the team as we can see in the Table 5.

| Person | Category | Cost per hour | Hours | Total |
|---|---|---|---|---|
| **Daniel Segovia Magaz** | Newly qualified | 23,43 € | 350 | 8.200,5 € |
| **Aurelio Ponz Vila** | PhD | 35,33 € | 20 | 706.6 € |

*Tabla 5. Wages*

So, the cost of the manpower is **8907.1 €**

In conclusion, the total budget of the project is **8.950,82 €.**

## Conclusion and future work

Inspections and repairs of the railway tracks is a hard work that needs a displacement of the machinery needed, besides of the human displacement who manner that machinery. Moreover, is a process which needs lot of time. This project intends to make that task easier and efficient, through the detections of cracks in the railroad track with the path of the railway itself.

Even though the project has been developed, it couldn't be integrated in FerroDron, so in a future work it would be the first task to solve. The data collected was with a test in a car, that can absorb the vibrations and the data might not be exact. What is true is that the vibration peak will be produced in the same way when the railway vehicle would find a crack in the railroad track.

On the other hand, an alternative sensor could be Pixhawk [37]. It is an open hardware project often used in drone control. Pixhawk has two IMUs, that would replace the sensor used in this project. Instead of two sensors like those which were used in this work, a Pixhawk will be installed in the platform of the railway vehicle. But in this way, the chassis absorb the vibrations and it could not provide real data. However, with a Pixhawk we can collect data from all the vehicle and not from each train rail independently , like we have done in the project.

In conclusion, this first version of the project has certain improvements, but the objectives have been fulfilled through collecting data from the sensor and publishing them in ROS topics. Besides, an analysis about where is a crack according to the vibrations graphics has been done.

# Anexos

## Anexo 1 : Código para la lectura de valores iniciales

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
MPU6050 accelgyro;
int16_t ax, ay, az;
int16_t gx, gy, gz;

#define LED_PIN 13
bool blinkState = false;

void setup() {
  Wire.begin();
  Serial.begin(38400);
  Serial.println("Inicializando dispositivos I2C...");
  accelgyro.initialize();

  Serial.println("Probando conexiones del dispositivo...");
  Serial.println(accelgyro.testConnection() ? "MPU6050 conexion establ
ecida" : "MPU6050 conexion fallida");

  pinMode(LED_PIN, OUTPUT);

  Serial.print("Iniciando tarjeta microSD:");
  if (!SD.begin(9)) {
    Serial.println("No se ha podido iniciar");
  }
  Serial.println("Iniciado correctamente");
}

void loop() {
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  Serial.print("a/g:\t");
  Serial.print(ax);
```

```
Serial.print("\t");
Serial.print(ay);
Serial.print("\t");
Serial.print(az);
Serial.print("\t");
Serial.print(gx);
Serial.print("\t");
Serial.print(gy);
Serial.print("\t");
Serial.println(gz);

Datos = SD.open("valoresiniciales.txt", FILE_WRITE);

if (Datos) {

        Datos.print("a/g:\t");
        Datos.print(ax);
        Datos.print("\t");
        Datos.print(ay);
        Datos.print("\t");
        Datos.print(az);
        Datos.print("\t");
        Datos.print(gx);
        Datos.print("\t");
        Datos.print(gy);
        Datos.print("\t");
        Datos.println(gz);

      Datos.close();

}
else {
  Serial.println("Error al abrir el archivo");
}

delay(10);
```

```
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
  }
```

## Anexo 2: Código para la lectura de valores escalados

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
MPU6050 accelgyro;
int16_t ax, ay, az;
int16_t gx, gy, gz;

#define LED_PIN 13
bool blinkState = false;

void setup() {
  Wire.begin();
  Serial.begin(38400);
  Serial.println("Inicializando dispositivos I2C...");
  accelgyro.initialize();

  Serial.println("Probando conexiones del dispositivo...");
  Serial.println(accelgyro.testConnection() ? "MPU6050 conexion establ
ecida" : "MPU6050 conexion fallida");

  Serial.print("Iniciando tarjeta microSD:");
  if (!SD.begin(9)) {
    Serial.println("No se ha podido iniciar");
  }
  Serial.println("Iniciado correctamente");

  pinMode(LED_PIN, OUTPUT);
}

void loop() {
   accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  float ax_m_s2 = ax * (9.81/16384);
  float ay_m_s2 = ay * (9.81/16384);
  float az_m_s2 = az * (9.81/16384);
  float gx_r = gx / 131;
```

```
float gy_r = gy / 131;
float gz_r = gz / 131;

Serial.print("a/g:\t");
Serial.print(ax_m_s2);
Serial.print("\t");
Serial.print(ay_m_s2);
Serial.print("\t");
Serial.print(az_m_s2);
Serial.print("\t");
Serial.print(gx_r);
Serial.print("\t");
Serial.print(gy_r);
Serial.print("\t");
Serial.println(gz_r);

Datos = SD.open("valoressi.txt", FILE_WRITE);

if (Datos) {

        Datos.print("a/g:\t");
        Datos.print(ax_m_s2);
        Datos.print("\t");
        Datos.print(ay_m_s2);
        Datos.print("\t");
        Datos.print(az_m_s2);
        Datos.print("\t");
        Datos.print(gx_r);
        Datos.print("\t");
        Datos.print(gy_r);
        Datos.print("\t");
        Datos.println(gz_r);

      Datos.close();

}
```

```
    else {
      Serial.println("Error al abrir el archivo");
    }

    delay(10);

    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
}
```

## Anexo 3: Código para la implementación de ROS

```
#include <ros.h>
#include <std_msgs/Float64.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"

ros::NodeHandle  nh;

std_msgs::Float64 ax_msg,ay_msg,az_msg,gx_msg,gy_msg,gz_msg;

ros::Publisher ax_pub("axx", &ax_msg);

ros::Publisher ay_pub("ayy", &ay_msg);

ros::Publisher az_pub("azz", &az_msg);

ros::Publisher gx_pub("gxx", &gx_msg);

ros::Publisher gy_pub("gyy", &gy_msg);

ros::Publisher gz_pub("gzz", &gz_msg);


MPU6050 accelgyro;
int16_t ax, ay, az;
int16_t gx, gy, gz;

#define LED_PIN 13
bool blinkState = false;

void setup() {
  Wire.begin();
  nh.initNode();
  nh.advertise(ax_pub);
  nh.advertise(ay_pub);
```

```
  nh.advertise(az_pub);
  nh.advertise(gx_pub);
  nh.advertise(gy_pub);
  nh.advertise(gz_pub);
  accelgyro.initialize();
  Wire.begin();

  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  float ax_m_s2 = ax * (9.81/16384);
  float ay_m_s2 = ay * (9.81/16384);
  float az_m_s2 = az * (9.81/16384);
  float gx_r = gx / 131;
  float gy_r = gy / 131;
  float gz_r = gz / 131;

  ax_msg.data = ax_m_s2;
  ay_msg.data = ay_m_s2;
  az_msg.data = az_m_s2;
  gx_msg.data = gx_r;
  gy_msg.data = gy_r;
  gz_msg.data = gz_r;

  ax_pub.publish( &ax_msg );
  ay_pub.publish( &ay_msg );
  az_pub.publish( &az_msg );
  gx_pub.publish( &gx_msg );
  gy_pub.publish( &gy_msg );
  gz_pub.publish( &gz_msg );

  nh.spinOnce();
  delay(100);
```

```
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
  }
```